Change Detection in Dynamic Attributed Networks

A thesis submitted in partial fulfilment of the requirements for the Degree of
Doctor of Philosophy in Statistics

Author:
Isuru Udayangani Hewapathirana

Supervisors:
Dr. Dominic Lee
Assoc Prof. Elena Moltchanova
Dr. Jeanette McLeod
Mr. Roger Jarquin

School of Mathematics and Statistics

University of Canterbury

February 2017

Abstract

A network provides a powerful means of representing complex relationships between entities by abstracting entities as vertices, and relationships as edges connecting vertices in a graph. Beyond the presence or absence of relationships, a network may contain additional information that can be attributed to the entities and their relationships. Attaching these additional attribute data to the corresponding vertices and edges yields an attributed graph. Moreover, in the majority of real-world applications, such as online social networks, financial networks and transactional networks, relationships between entities evolve over time. This dynamic behaviour can be modelled using a time sequence of attributed graphs. Change detection in dynamic attributed networks is an important problem in many areas, such as fraud detection, cyber intrusion detection and health care monitoring. It is a challenging problem because it involves a time sequence of attributed graphs, each of which is usually very large and can contain many attributes attached to the vertices and edges, resulting in a complex, high dimensional mathematical object.

Spectral embedding methods provide an effective way to transform a graph to a lower dimensional latent Euclidean space that preserves the underlying structure of the network. Although change detection methods that use spectral embedding are available, they do not address sparsity and degree heterogeneity that occur in most real-world graphs. Furthermore, a majority of these methods focus on changes in the behaviour of the overall network. This is a drawback for applications where it is more important to detect changes in entity behaviour rather than the behaviour of the overall network.

In this thesis, we propose a novel concept of applying Procrustes techniques to embedded points for vertices in a graph to detect changes in entity behaviour. By adapting previously developed techniques in spectral graph theory, we first formulate a strategy to extract an embedding from the weighted adjacency matrix of each graph in the sequence. Our spectral embedding approach not only addresses sparsity and degree heterogeneity issues, but also obtains an estimate of the appropriate embedding dimension. Each embedded point represents the behaviour of an entity at a given time instant. We employ a Procrustes analysis procedure to compare embeddings across time and to quantify changes. We call this method CDP (change detection using Procrustes analysis). We demonstrate the performance of CDP for undirected, unipartite graphs through extensive simulation experiments. CDP successfully detects various types of vertex-based changes considered in the experiments, including (i) changes in vertex degree, (ii) changes in community membership of vertices, and (iii) unusual increase or decrease in edge weight between vertices. Furthermore, when applied to the dynamic network for the Enron email dataset, CDP successfully detects, at the appropriate times, several fraudsters who were known to be involved in the scandal. For both simulation and

real-world experiments, we compare the change detection performance of CDP with two other baseline methods (the method in Idé and Kashima (2004) and improved version) that employ alternative spectral embedding approaches. In both cases, CDP generally show superior performance.

A multi-view network depicts multiple types of relationships between a set of entities, that may either be obtained from the same source or from different sources. A dynamic multi-view network can be represented as a time sequence of three-dimensional weighted adjacency tensors. The second part of this thesis extends CDP to handle multi-view change detection based on tensors. We propose two methods to obtain embedded points for vertices. The first method, MCDP-I (multi-view change detection using Procrustes analysis-I), does this by using higher order singular value decomposition to factorize a tensor. The second method, MCDP-II (multi-view change detection using Procrustes analysis-II), first embeds each slice of the tensor separately using matrix factorization, and then employs Procrustes techniques to calculate an average embedding. We systematically investigate the performance of MCDP-I and MCDP-II based on several simulation experiments. In all experiments, the change detection performance of our methods are compared against five other methods (Han et al., 2015; Liu et al., 2013; Tang et al., 2012) that differ from each other mainly in the way that information from different tensor slices are combined to give an embedding for the vertices. Both MCDP-I and MCDP-II successfully detect all types of vertex-based changes considered in the experiments. MCDP-I generally performs better than MCDP-II when the change information contained in the tensor is consistent across all slices; otherwise, MCDP-II is generally better. When applied to the multi-view Enron email network, MCDP-II successfully detects several known fraudsters while MCDP-I detects only one of them.

The novel concept of applying Procrustes techniques to embedded points for vertices introduces a new direction for change detection. The algorithms we have developed can be applied to different areas, such as cyber-intrusion detection, fraud detection, healthcare monitoring and detection of natural disturbances in the eco-system.

Acknowledgements

First and foremost, I pay my homage to Lord Buddha, Dhamma, and Sangha.

This thesis becomes a reality with the kind support and help of many important individuals. I would like to extend my sincere thanks to all of them. First, I would like to express my gratitude to all my supervisors for their unwavering support, guidance and insight throughout this research project. My deepest thanks go to Dr. Dominic Lee for his tremendous support in limitless ways. I still remember, during the early stages of my PhD when I was new to Christchurch, he was always there to help me and my family. He was a great strength for me throughout all these years and never let me down when I needed his help. I am utterly thankful for his excellent supervision and valuable advice that immensely improved my academic as well as professional skills and taught me how to become a good researcher. I am also grateful to Associate Professor Elena Moltchanova for her guidance throughout this research project. Her constructive reviews of my work, taught me how to express my ideas in a clear and precise manner. I also thank Dr. Jeanette McLeod for helping me strengthen my knowledge in Graph Theory. Furthermore, I owe her many thanks for reading, proofing and commenting on my writing that greatly helped me to improve. I would also like to thank Mr. Roger Jarquin for providing his support and guidance for the progress of this research project.

My sincere thanks go to all the staff and postgraduate colleagues of the School of Mathematics and Statistics at the University of Canterbury, for providing me a friendly and enjoyable platform to work smoothly throughout these three years. I must specially thank Dr. Raazesh Sainudiin for being my mid-PhD examiner and also letting me sit in his scalable data science course, that greatly helped me to develop my skills as a data scientist. I must also thank Paul Brouwers and Steve Gourdie for their friendly and prompt assistance during my computer related problems. I am also thankful to Penelope Goode for her friendly and approachable manners whenever administration related help was needed.

I am grateful to the Wynyard group, New Zealand, for awarding me a scholarship for the entire three years, without which I would never have been able to pursue a PhD at the School of Mathematics and Statistics at the University of Canterbury. It is my pleasure to thank all the people in the data analytics team at Wynyard for their helpful advice, suggestions and comments on my work that I presented to them. My special thanks go to Dr. Shakira Suwan for not only supporting me in research work, but also being a great colleague and friend since the beginning of my study.

Last, but not least, I thank my family for their unconditional love and support: I thank my parents for directing me to pursue my studies in statistics and facilitating me with

everything I need to excel in it. I am thankful for my sister and brother-in-law for always being there for me whenever I needed them. I would also like to thank my mother-in-law and father-in-law for constantly caring for me and looking out for my needs. My deepest thanks go to my husband Charaka and my daughter Thimethya. I am utterly grateful for the sacrifices they have made to help me. It is their love, patience and support that allowed me to pursue my research freely and contentedly throughout all these years.

Contents

List of Figures

List of Tables

Abbreviations

CCA	canonical correlation analysis
CDP	change detection using Procrustes analysis
DCSBM	degree corrected stochastic block model
DMR	dimension by matrix reconstructon
EVD	eigen-value decomposition
GMM	Gaussian mixture model
GPA	generalized orthogonal Procrustes analysis
HOSVD	higher order singular value decomposition
LSM	latent space model
MCDP	multi-view change detection using Procrustes analysis
RDPG	random dot product graph
SBM	stochastic block model
SVD	singular value decomposition

Notations

$\mathbb{R}^d$	set of all real numbers in dimension d
$\mathbb{R}^{n \times d}$	array of all real numbers in dimension $n \times d$
$\mathbb{Z}_+$	set of positive integers
G	Graph set which consists of an order pair (V, E)
V	Set of vertices in G
E	Set of edges in G
$\|V\| = n$	Graph order or number of vertices in G
A	$n \times n$ binary adjacency matrix
W	$n \times n$ weighted adjacency matrix
D	$n \times n$ diagonal matrix of vertex degrees
I	Identity matrix
$\langle \mathbf{x}, \mathbf{y} \rangle$	dot product of two vectors
$\| \, . \, \|_F$	Frobenius norm
$X \perp Y$	orthogonality
$\| \, . \, \|_2$	L_2 norm
$\mathbb{P}[B]$	probability of event B
$X \sim f$	random variable X has density f
$f(x\|y)$	probability density of x conditioned on y
$\|a\|$	absolute value of scalar a
$a \ll b$	a is much less than b
$\log(x)$	the logarithm of x to base $e = 2.718281828459$

Dedicated to my beloved family ...

Chapter 1

Introduction

1.1 Motivation

Rapid developments in the field of information technology have resulted in a proliferation of datasets over the past few decades. Examples include supermarket transaction data, credit card records, medical records, and records of telephone call information. The availability of large datasets led to a high demand for new methods to extract useful information. Thus the field of *data mining* started to develop. Data mining involves methods for exploring, analysing, and modelling large datasets to find valid, novel, and interpretable patterns (Fayyad et al., 1996). The area of data mining which attempts to discover unusual occurrences in dynamic (time-varying) datasets is called *change detection*. Detecting changes provides early warnings of events that can have serious impact. For example, early detection of an abrupt change in the stock market could help improve decision making and avoid possible losses for businesses due to wrong decisions and weak trading systems. Also, being able to detect a possible failure in an aircraft system could save thousands of lives as well as millions of dollars. The majority of traditional data mining methods assume entities in a dataset to be independent, and ignore the natural relationships among them (Huang and Gao, 2014). However, entities in most real-world datasets are not independent, but are related to each other in different ways. For accurate change detection, it is vital to consider these underlying relationships. For example, the extent to which a transaction is fraudulent in credit card data can be identified more accurately by considering the usual purchasing behaviour of the card-holder as well as the usual purchasing behaviours of other card holders who bought the same product (Singh and Singh, 2015). The need for regarding such inherent dependencies among entities has aroused interest for network-based change detection methods.

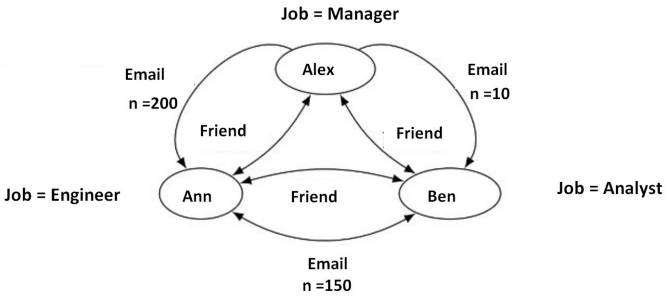


FIGURE 1.1: Illustration of an attributed network.

A network is a collection of entities, that have inherent relationships. Some examples include a social network of friendships among people, a communication network of company employees connected by phone calls, emails or text messages, and a biological network of neurons connected by their synapses. A network can be mathematically conceptualized as a graph by associating entities with vertices, and relationships with edges connecting vertices in the graph. For example, in the graph representation of a social network like Facebook, vertices may represent friends and edges represent friendship connections. It is also possible to capture more information by attaching additional attribute data to the vertices and edges, resulting in an attributed graph. An illustrative example of a small attributed graph of three vertices is shown in Figure 1.1. Each vertex is tagged with attribute information about the job role of the corresponding employee, while the edges are tagged with attribute information based on the type of relationship (whether the relationship is based on friendship or email sending patterns). An edge can be further weighted to quantify the corresponding relationship. In the example given in Figure 1.1, the edges representing email sending patterns are weighted to represent the number of emails.

Most real-world networks evolve as time progresses. That is, the entities and their relationships keep evolving with time. This type of relational data can be represented as a dynamic network. For example, a communication network of a company is a dynamic network because new employees (entities) join the network and communication patterns (relationships) are modified continuously. Although both the entities and the relationships in a network can vary over time, for the purpose of this thesis, we assume that a dynamic network consists of a fixed set of entities with time varying relationships between them. A dynamic network can be represented as a time sequence of graphs, each representing the entities (as vertices) and their relationships (as edges) at a given time instant.

Change detection in dynamic networks is the process of continuously monitoring a network for deviations in the behaviour of entities and their relationship structure. Figure 1.2 illustrates this based on a toy example. The graph obtained at the current time

instant represents the current behaviour of entities and their relationships, while the sequence of graphs inside the window containing the *recent past* time instants, characterise *profile* behaviour. A general change detection method aims to build a model based on the profile behaviour of the graph during the recent past, and search for deviations in the behaviour of the current graph with respect to the model. The majority of change detection methods monitor a *representative summary* extracted from the overall graph, or parts of the graph, such as vertices, edges or subgraphs[1] to characterise behaviour and analyse deviations. For example, in order to detect fraudulent collaborations in social networks, Yu et al. (2014) track the properties of communities (a special type of subgraph which will be explained in Chapter 2), while Neil et al. (2013) observe the appearance of unlikely edges to detect intrusions in a computer network. The choice of unit (either overall graph, vertices, edges or subgraphs) to track over time to detect changes, mainly depends on the given application.

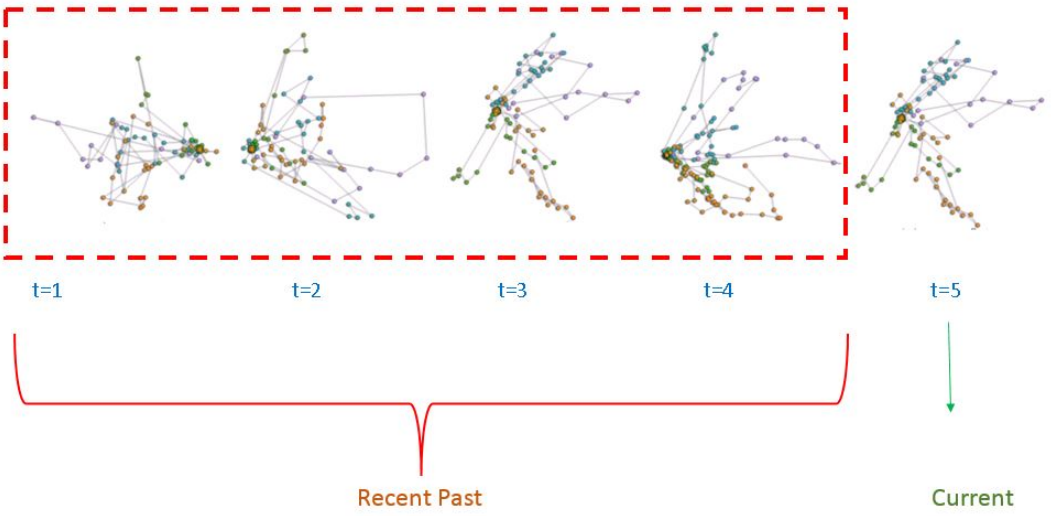


FIGURE 1.2: **Illustrative example of change detection**. Each graph represents the behaviour of the entities and their relationships at the corresponding time instant. A sequence of graphs inside the window containing the recent past time instants characterises profile behaviour of entities and their relationships. A change detection method compares the current behaviour of the overall graph, vertices, edges, or subgraphs with their profile behaviour.

Change detection in dynamic networks has many vital applications:

- Social network analysis:
  Changes in popular social networks such as Facebook, Twitter, LinkedIn or Youtube may arise as a result of individuals or groups of individuals making deviations in their usual relationship patterns. For example, a group of fraudsters in an on-line auction network who previously had no relationships may suddenly collaborate to boost the reputation of their product. Such changes will be reflected as changes

[1] A subgraph is a subset of vertices and edges in the graph.

in the corresponding dynamic network, so change detection enables detection of those fraudsters (Pandit et al., 2007).

- Financial market analysis:
  A financial network represents the dynamic inter-relationships between the world's various financial markets. Changes in international trade or countries' stock markets at a certain point in time, correspond to changes in the structure of the corresponding financial network (Durante and Dunson, 2014). Such changes should be detected early to minimize financial crisis situations.

- Network traffic monitoring:
  A dynamic computer network can be represented as a time sequence of edge attributed graphs, where the vertices represent the computers, and the edges represent the volume of traffic between each pair of computers. A change in the usual amount of traffic between a pair of computers may indicate a fault in the system or even an intrusion. Such changes can be detected by continuously tracking the edges over time (Sun et al., 2006).

- Analysis of earth-science data:
  Earth scientists detect natural disasters by analysing data about variables such as vegetation cover, temperature, and pressure at certain locations over time. Natural disasters such as wildfires correspond to abrupt changes in the time-series of these variables. However, these variables are most often inherently correlated with each other. Accounting for these inherent correlations, by formulating a network, and applying network-based change detection methods, increase the detection ability of such natural disasters (Cheng et al., 2008).

- Connectomic applications:
  Connectomics involves the study of inter-relationships among different regions of the brain. Inter-relationships between brain regions can be observed over time, and represented as a dynamic network. A sudden neurological failure such as an epilepsy attack, will correspond to a change in the structure of the network (Durante and Dunson, 2014). Hence, change detection can be used to improve diagnosis and treatment of such medical conditions.

In order to accurately detect changes, we need to consider the following issues:

Scale and dynamics: Many real-world networks are considerably large in size. For example, the Facebook contains more than one billion users,[2] and the World Wide

[2] http://newsroom.fb.com/company-info/

Web consists of more than 40 billion web pages[3]. A graph representing the entity-relationships in these datasets is also large, with billions of vertices and edges. If such data are collected in real time, the speed at which they arrive is also an issue. We require fast methods to study the structure of such graphs at each time instant.

Complexity: As mentioned previously, the vertices and edges of a graph can be tagged with attributes capturing specific information about entities and their relationships. A change detection method should effectively combine all available and useful information in order to obtain accurate results. However, the inclusion of edge and vertex attributes increases the complexity.

Lack of prior information: Data we receive often arrives with no a priori knowledge about the underlying situation. Hence, we have no knowledge about the recent past behaviour, and this should be obtained from the data using a method that is both accurate and computationally efficient (Nickel, 2013).

This thesis uses a spectral-based strategy to address the above challenges, and detect vertex-based changes in a time sequence of graphs.

1.2 Thesis Overview

In this thesis, we consider relationships between entities to be symmetric. Hence, we only deal with undirected dynamic networks, which are mathematically represented as a time sequence of graphs with undirected edges (a more detailed description is given in Section 2.1). We focus on *off-line change detection*, that is, we consider the number of time instants to be fixed and non-increasing. Our definition of change detection is as follows:

Definition 1.1 (Change Detection in Dynamic Networks)**.** Given a dynamic network represented as a sequence of graphs, find vertices whose behaviours are different from the recent past. If necessary, also combine the amount of change calculated for all vertices to quantify change in the overall graph.

In order to detect changes in a dynamic network, we perform the three steps defined below:

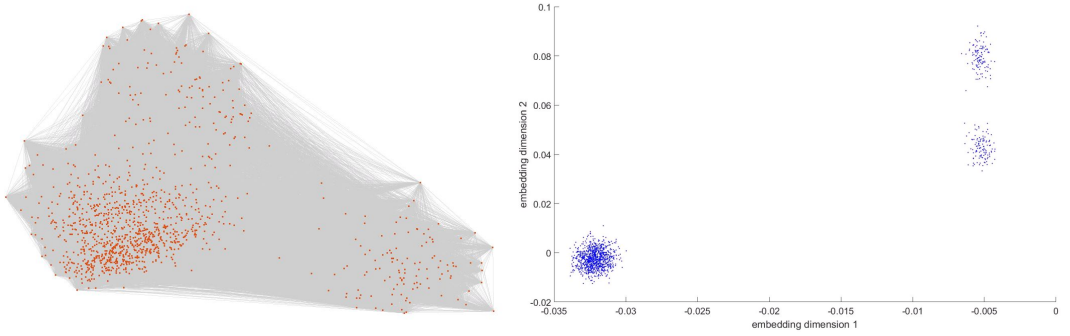
1. *Feature* extraction: A feature provides a compact and interpretable representation of the behaviour of a vertex in the graph at each time instant.

[3] http://www.worldwidewebsize.com/

2. *Profile feature* calculation: A profile feature represents the average behaviour of a vertex in the graph during the recent past. Hence, it can be obtained by combining the features obtained for the recent past time instants.

3. *Change score* calculation: A change score for a vertex quantifies the amount of change in its behaviour at the current time instant compared to the recent past. Thus, it is obtained by evaluating the dissimilarity between the current feature and the profile feature.

Extracting vertex features at each time instant is a challenge because of the scalability and complexity issues mentioned in Section 1.1. Spectral embedding methods provide an elegant way to address these challenges and extract vertex features via matrix or tensor factorizations (extensively discussed in Section 2.5). The term *embedding* means that vertices in a graph are mapped into a lower dimensional latent Euclidean space. The embedded points in the latent space provide a simpler representation of the underlying relationship structure of the network (Figure 1.3).



(A) A network of 1000 vertices

(B) Vertices in the embedded space

FIGURE 1.3: **An example of an embedding of a graph**. The original graph with 1000 vertices and millions of edges is shown in Figure 1.3 (A). Figure 1.3 (B) shows the scatter plot of the first two dimensions of the corresponding embedding of the graph in Figure 1.3 (A). We can observe three clear groups of vertices in the embedded space, which are hard to distinguish in the original graph representation.

Although there exist several methods that exploit spectral methods for change detection (Akoglu and Faloutsos, 2010; Hirose et al., 2009; Idé and Kashima, 2004), they are mostly focused on detecting changes in the densely connected regions of a graph. These methods generally neglect changes in less connected or sparse regions of the graph. Furthermore, there are not many spectral-based change detection approaches that take advantage of multiple edge attributes. Methods such as Sun et al. (2006) that do exploit multiple edge attributes, mainly focus on detecting changes involving the overall network. In our work, we employ a spectral embedding approach that handles sparseness as well as multiple edge attributes to obtain vertex features.

At each time instant, calculation of change scores involves evaluating the dissimilarity between embeddings that represent the recent past and the current time instant. In this thesis, we employ a statistical technique called Procrustes analysis for obtaining both the profile features and the change scores. Procrustes methods (a detailed description is given in Section 2.6) are commonly used for shape recognition and object matching in video processing applications (Vaswani et al., 2005). A window is moved through the whole time sequence of embeddings to obtain profile features and calculate change scores at each time instant. Hence, our method can be called a *moving window approach*.

Additionally, change detection can be followed by *outlier* or *anomaly* detection. An outlier or anomaly is a data point that deviates so much from other data points, that it arouses suspicion as to whether it was generated by a different mechanism (Hawkins, 1980). An anomaly detection method can be applied to map each change score to $\{0, 1\}$ or $[0, 1]$; The mapping to $\{0, 1\}$ indicates whether a vertex is an anomaly (indicated by 1) or not (indicated by 0) (Idé and Kashima, 2004), while the mapping to $[0, 1]$ gives the probability of a vertex being an anomaly (Heard et al., 2010). A comprehensive summary on numerous anomaly detection methods can be found in Chandola et al. (2009). However, the focus of this thesis is mainly on the change detection approach that obtains change scores at each time instant.

1.2.1 Thesis Objectives

Given a time sequence of undirected, edge-attributed graphs, our main objective is to propose a method to detect vertices that show change from profile behaviour. We divide this main objective into two sub-objectives:

(i) to detect changes involving vertices in a time sequence of graphs, based on matrix-based spectral embedding methods, and
(ii) to extend (i) for multi-graphs represented as three-dimensional tensors.

These two objectives are discussed in chapters 3 and 4, respectively.

1.2.2 Thesis Contributions

In this thesis, we explore methods which detect vertex-based changes in a time sequence of edge-attributed graphs. The principal contributions of this thesis are as follows:

In the first part of this thesis, we formulate a novel method to detect changes involving vertices by representing the time sequence of graphs as a time sequence of weighted adjacency matrices. By incorporating previously developed concepts in spectral graph

theory (Amini et al., 2013; Chung, 1997; Le et al., 2015), each weighted adjacency matrix is transformed into a low dimensional embedding that gives the features representing the behaviour of all vertices at a given time instant. Our spectral embedding approach that addresses sparsity and degree heterogeneity in graphs is one key contribution that distinguishes our work from previous spectral embedding-based change detection approaches. The basic idea of our change detection procedure is, if a vertex has changed, then its feature should be different from its features obtained during the recent past. Thus, we obtain change scores by calculating the dissimilarity between embeddings that represent the recent past and the current time instant. The concept of comparing two sets of embeddings using orthogonal Procrustes matching was initially introduced by Tang et al. (2014). In this thesis, we adapt their ideas and formulate a generalized orthogonal Procrustes technique to obtain profile features and calculate change scores for vertices. The novel application of Procrustes techniques for change detection in dynamic networks is another important contribution of our research.

The performance of the new method is assessed using extensive simulation experiments and one real-world dataset. In all situations, our method shows good change detection performance that is substantially better than the baseline methods.

The second part of this thesis extends our new change detection method to exploit multiple edge attributes represented by tensors. We achieve this by extending our matrix based spectral embedding strategy in two main ways to obtain an embedding from a tensor. Our first extension follows a tensor factorization approach to obtain a single embedding that jointly summarizes the connectivity structure represented by all slices of the tensor. Our second extension separately obtains an embedding from each slice of the tensor and combines them using generalized Procrustes techniques. We systematically investigate the performance of the extended change detection methods using simulation experiments and a real-world dataset. The results show good performance for both methods in detecting various types of changes. However, the second method is the most reliable in the presence of conflicting information across slices.

1.2.3 Structure of Thesis

Chapter 2 (Background Material) provides a literature review on the underlying methodologies that we employ in this thesis. We discuss some random graph models, some currently existing change detection methods, pertinent spectral embedding methods, and Procrustes analysis techniques that we use in our work.

Chapter 3 (Change Detection in Dynamic Networks using Procrustes Analysis) presents a novel method to detect changed vertices in a time sequence of graphs,

represented as a time sequence of weighted adjacency matrices. We use the spectral embedding techniques and the Procrustes analysis techniques discussed in Chapter 2 to extract a representative summary from each time instant, and calculate vertex change scores respectively. We employ random graph modelling techniques discussed in Chapter 2 to generate synthetic dynamic networks in order to evaluate the performance of our method. In our experiments we compare the performance of our proposed method with another spectral-based change detection method and a modified version of that method. We further apply our method to the Enron email dataset to detect fraudsters involved in the famous Enron scandal.

Chapter 4 (Change Detection in Multi-view Dynamic Networks using Procrustes Analysis) extends the change detection method developed in Chapter 3 to handle multiple edge attributes represented using tensors. In this chapter, we propose two main spectral embedding strategies. Using simulation experiments, we compare our proposed methods to several other baseline methods that use different ways of obtaining an embedding from a tensor. We assess performance with respect to change detection. We show the advantages and disadvantages of each method, and finally present the best method to use in change detection. We further apply our proposed change detection methods to the Enron email dataset.

Chapter 5 (Concluding Remarks) summarizes the outcomes and importance of our overall work. We also provide a discussion of possible directions for future research.

Chapter 2

Background Material

2.1 Types of Graphs

Relationships between entities can often be represented as a network. Mathematically, we represent a network as a graph, which is a collection of vertices connected by edges. The type of information represented and structures allowed will dictate the types of graphs we use.

If we consider only the absence or presence of a relationship between two entities, the resulting graph is a *simple graph* where there is at most one edge between any pair of vertices. These graphs do not contain edges connecting a vertex to itself (also known as self-loops). A *weighted* or *attributed graph* allows the association of information about entities and relationships with both vertices and edges, respectively. For example, for an attributed social network, each vertex can be tagged with additional information such as name and designation of the associated entity, and each edge tagged with information pertaining to the strength of the relationship or frequency of interaction. Compared to a simple graph with binary edges, an edge-attributed graph provides more information about the relationships between entities in the network (Bacry et al., 2015).

The edges of a graph can be either directed or undirected. A graph with directed edges represents asymmetric relationships in the underlying network, while a graph with undirected edges considers all entity relationships to be symmetric. For example, a network of email users can be represented as a directed graph with edge directions depicting an email sent from one user to another. However, by simply ignoring the directions of the relationships, the same network can be represented as an undirected graph. A graph consisting of vertices of the same type is a *unipartite graph*. When vertices in the graph are of k different types, and edges exist only between vertices of

different types, we obtain a *k-partite graph*. The *bipartite graph* ($k = 2$) is common in many real-world applications (Sun et al., 2007; Tong and Lin, 2011). For example, we can use a bipartite graph to represent the relationships between authors who publish papers at a conference, where the authors and conferences are the two types of vertices, and an edge is present if an author participates in a given conference.

In most real-world phenomena, we have multiple sources of information describing different types of relationships associated with entities, which can be captured by a *multi-view network*. For example, we can build a multi-view network over a given set of people based on their relationships over multiple social networking sites such as Facebook, Twitter and Youtube (Tang et al., 2012). A multi-view network can also be employed to capture a network obtained from one information source, but with various relationship types. For example, a relationship in the Twitter network can represent users whom they follow (or who follow them), users whom they re-tweet (or re-tweet them), and users whom they mention in their posts (or who mention them) (Greene and Cunningham, 2013). A multi-view network is mathematically represented as a *multi-graph*, which is a graph with multiple edges. However, we can simplify the graph and represent each of these edges by an edge attribute on a single edge. In this case, each edge attribute corresponds to a different view of the network. In the network analysis literature, multi-view networks are also referred to as multi-dimensional networks (Tang et al., 2012), multi-layer networks (Papalexakis et al., 2013), multi-relational networks (Cai et al., 2005), and multiplex networks (Liu et al., 2013).

The networks discussed so far are static networks. Dynamic networks are networks that evolve over time. In order to study the evolving behaviour of dynamic networks, they are sometimes represented as a sequence of graphs (Idé and Kashima, 2004; Priebe et al., 2005). Each graph represents the relationships between the entities in the dataset at a specific time instant. This thesis will focus on dynamic networks. In practice, both the entities and relationships in a dynamic network can vary over time; however, in this thesis, we will limit our attention to dynamic networks where there is a fixed set of entities and only the relationships between them vary over time.

2.2 Graph Theoretic Notations and Concepts

A network can be mathematically represented as a graph, $G = (V, E)$, where V is a set of n *vertices* and E is a set of *edges*. For example, in the case of a simple graph with n vertices, the set E will contain a list of those edges that are present in the graph. From this edge list, the presence or absence of an edge between each pair of vertices can be encoded as an $n \times n$ binary adjacency matrix, A, where each element,

$A_{i,j}$, is 1 if the edge between vertices i and j appears in E, and 0 otherwise. In a weighted graph, additional information is assigned to the edges or vertices or both, this information is called a *weight*. A weighted graph can be encoded as an $n \times n$ weighted adjacency matrix, W, and $W_{i,j}$ is the non-negative weight of the edge between vertices i and j, and $W_{i,j} = 0$ means that vertices i and j are not connected by an edge. In this thesis, we consider graphs where the edge weights are proportional to the strength of the relationship between the corresponding entities.

A dynamic network can be mathematically represented as a time sequence of graphs, $G^1, G^2, \ldots$, where $G^t = (V^t, E^t)$. Since our focus is on a dynamic network with a fixed set of entities, we will have $G^t = (V, E^t)$ as a special case.

The graph theoretic concepts that are used in this thesis are defined below. The following definitions are for undirected graphs for which the adjacency matrix or weighted adjacency matrix is symmetric. Definitions that involve A are also applicable to W, but are stated below for A only.

- Order: The number of vertices in the graph, $|V| = n$.
- Degree of a vertex: The number of edges connected to it, or, in a weighted graph, it is the sum of the weights of the edges connected to it (Newman, 2004a). In terms of A, the degree of vertex i is defined as

$$d_i = \sum_{j=1}^{n} A_{i,j}. \tag{2.1}$$

- Size: The number of edges in the graph.
- Centrality: Centrality measures a vertex's importance in the graph. For example, *degree centrality* of a vertex provides local information about its importance (Salter-Townshend et al., 2012). Since the maximum vertex degree is $n - 1$, the standardized degree centrality of vertex i is defined as

$$C_d(i) = \frac{d_i}{n-1}. \tag{2.2}$$

  Higher degree centrality indicates higher importance of the corresponding vertex.
- Path: An alternating sequence of vertices and edges, where each edge is incident to the pair of vertices that are immediately before and after it in the sequence.
- Length of a path: For an unweighted graph, the length of a path is the number of edges in the path. For a weighted graph, it is the sum of the reciprocal of weights of the constituent edges in the path.

- Shortest path: The path of minimum length between two vertices.
- Diameter: The longest shortest path between any two vertices in the graph.
- Connected component: A subset of vertices for which there is a path between any two vertices in the subset. A graph is *connected* if the whole graph is one connected component.
- Subgraph: A graph $G' = (V', E')$ is a subgraph of G, if and only if, $V' \subseteq V$, and $E' \subseteq E$.
- Sparse: A graph is defined as sparse, if its average vertex degree, $\frac{1}{n}\sum_i^n d_i$, is less than five. This is the heuristic definition given in Amini et al. (2013). A graph is *dense* if its average vertex degree is much greater than five.

2.2.1 Quick Review of Linear Algebra

In this section, we provide a brief review of some necessary basic linear algebraic concepts.

- The maximum number of linearly independent rows (or columns) of an $n \times m$ matrix, A, is called its *rank*, denoted $\text{rank}(A) \leq \min\{n, m\}$.
- The rank of a diagonal matrix is the number of non-zero diagonal elements present.
- An $n \times n$ matrix A is orthogonal if A^{-1} exists and $A^{-1} = A^T$. Then, $AA^T = A^TA = I$.
- Let A be an $n \times n$ square matrix. A scalar λ and an $n-$dimensional vector, $\mathbf{x} \neq 0$ that satisfy
$$A\mathbf{x} = \lambda\mathbf{x}, \tag{2.3}$$
are called an *eigenvalue* and an *eigenvector* of the matrix A, respectively. The eigenvector corresponding to the largest eigenvalue is called the *principal eigenvector* of A.
- Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote the eigenvalues of A, which are all assumed distinct, and let $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ denote the matrix with eigenvectors of A, respectively. Then,
$$AX = X\Lambda, \tag{2.4}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}. \tag{2.5}$$

The eigenvectors corresponding to distinct eigenvalues are linearly independent. Thus, we can rewrite Equation 2.4 as

$$A = X\Lambda X^{-1}. \tag{2.6}$$

Equation 2.6 is called the *eigenvalue decomposition* of A. The number of non-zero eigenvalues of A is at most rank(A).

- For a symmetric matrix, A where $A = A^T$, the eigenvectors corresponding to distinct eigenvalues are *orthonormal*. Let X be the matrix with columns corresponding to the eigenvectors of A. Then, $X^{-1} = X^T$ and $XX^T = X^TX = I$. Thus, for a real and symmetric matrix[1], the eigenvalue decomposition can be written as

$$A = X\Lambda X^T. \tag{2.7}$$

- An $n \times n$ real symmetric matrix A is *positive definite* if $\mathbf{x}^T A\mathbf{x} > 0$ for all $n-$dimensional vectors, $\mathbf{x} \neq 0$; it is *positive semi-definite* if $\mathbf{x}^T A\mathbf{x} \geq 0$ for all $n-$dimensional vectors, $\mathbf{x} \neq 0$. In other words, when all eigenvalues of a symmetric matrix A are positive, A is a positive-definite matrix. When the eigenvalues are are all non-negative, A is said to be positive semi-definite.

- The *dot product* between two vectors, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$, is defined as

$$< \mathbf{x}, \mathbf{y} > = \mathbf{x}^T\mathbf{y} \tag{2.8}$$

$$= \sum_{i=1}^{n} x_i y_i. \tag{2.9}$$

- The *Euclidean norm* (also known as *Frobenius norm* or *length*) of vector $\mathbf{x} \in \mathbb{R}^n$ is defined as

$$\| \mathbf{x} \|_F = (|x_1|^2 + |x_2|^2 + \ldots |x_n|^2)^{1/2} \tag{2.10}$$

$$= (\sum_{i=1}^{n} |x_i|^2)^{1/2}. \tag{2.11}$$

[1] If A is both real and symmetric, then the eigenvalues of A are all real. Otherwise, eigenvalues can be complex, even if all entries of A are real.

- The L_2 norm of a matrix A is defined as

$$\| A \|_2 = \max_{\|\mathbf{v}\|_F=1} \| A\mathbf{v} \|_F . \tag{2.12}$$

- The *Kronecker product* of matrices, $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{K\times L}$, is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}, \tag{2.13}$$

  where the size of the resulting matrix is $(IK) \times (JL)$.

2.3 Random Graph Models

A random graph model defines a likelihood function for the network data, based on a probability model for the edges in the graph. The probability model can include information about vertex and edge attributes when available. We encounter different models in literature (Goldenberg et al., 2010; Salter-Townshend et al., 2012) that differ from each other based on the complexity of the models, the type of data modelled, as well as their ultimate inferential objectives (Ghosh et al., 2010). Below we review the models that are relevant to this thesis.

2.3.1 Erdös-Rényi Random Graph Model (ER model)

One of the most basic probabilistic models that describe a graph is the Erdös-Rényi Random Graph Model (Erdős and Rényi, 1959). The model describes an undirected graph based on the assumption that the probability of an edge occurring between vertices, i and j, is a fixed value, p, for all edges in the graph. In terms of the $n \times n$ binary adjacency matrix, A, the edges are assumed to be independent and identically distributed (i.i.d.) with $A_{i,j} = 1$ with probability p, and $A_{i,j} = 0$ with probability $1 - p$. The likelihood of a simple graph under the ER model is given by

$$\mathbb{P}[A|p] = \prod_{i\neq j} p^{A_{i,j}}(1-p)^{(1-A_{i,j})}. \tag{2.14}$$

The independence assumption on the edges simplifies the analysis of the graph, but it implies that the degree distribution of large graphs follows a Poisson distribution (Zaki and Meira, 2013). However, the degree distribution observed in real-world graphs is

not Poisson (Savage et al., 2014). Thus, the ER model is not suitable for analysing real-world graphs (Salter-Townshend et al., 2012).

The four most common properties of real-world graphs are *transitivity*, *homophily* by attributes, *clustering*, and *global dependencies*.

- *Transitivity* implies that if two vertices, i and j, are connected, and if vertices, j and k, are also connected, then vertices, i and k, have a higher probability of being connected to each other.
- *Homophily by attributes* means that vertices possessing similar attributes are more likely to be connected by an edge (McPherson et al., 2001). Homophily in graphs can be used to predict unknown relationships as well as unknown attributes (Nickel, 2013). For example, if a student's age is unknown in a social network, it can be predicted using his friends' ages.
- Vertices in a graph can be partitioned into groups or *clusters* such that members of the same group have similar connectivity patterns. This is called the *clustering property*. The clustering property can also be used to predict unknown edges between vertices when their cluster membership is known.
- *Global dependencies* refer to dependencies among vertices that persist across multiple views. For example, two students in the same university can also have connections as Facebook friends, room-mate friends, or lab partners. These global dependencies can be exploited to predict unknown relationships, as well as to detect clusters of entities with similar relationships.

It is often beneficial to use a random graph model that captures some or all of the above-mentioned properties of real-world graphs. The rest of Section 2.3 describes some models that satisfy this requirement.

2.3.2 Latent Space Model (LSM)

The *latent space model* by Hoff et al. (2002) assumes that each vertex, i, is associated with a latent random vector, $\mathbf{x}_i$, in a d-dimensional Euclidean space. The probability of an edge between two vertices, i and j, is conditional on: (i) the two latent positions, $\mathbf{x}_i$ and $\mathbf{x}_j$, (ii) available covariate information, $\mathbf{c}_{i,j}$, derived from vertex and edge attributes on the corresponding edge, and (iii) model parameters, Θ. In terms of the binary adjacency matrix, A, for a simple graph, the likelihood function of the model is given by

$$\mathbb{P}[A|X, C, \Theta] = \prod_{i \neq j} \mathbb{P}(A_{i,j}|\mathbf{x}_i, \mathbf{x}_j, \mathbf{c}_{i,j}, \Theta). \tag{2.15}$$

Each non-diagonal element in $A_{i,j}$ is sampled from a Bernoulli distribution. Hoff et al. (2002) introduced two latent position models; the distance model and the projection model. Let $p_{i,j}$ be the probability of an edge between vertices i and j, that is, $\mathbb{P}[A_{i,j} = 1|p_{i,j}] = p_{i,j}$. The distance model uses the logistic regression model to parametrize $\mathbb{P}[A|X, C, \theta]$ as

$$\begin{aligned} \log \frac{p_{i,j}}{1-p_{i,j}} &= \alpha + \beta^T \mathbf{c}_{i,j} - |\mathbf{x}_i - \mathbf{x}_j| \\ &= \eta_{i,j}, \end{aligned} \tag{2.16}$$

where $\mathbf{c}_{i,j}$ is a vector of covariates specific to vertex pair, i and j, $|\mathbf{x}_i - \mathbf{x}_j|$ denotes the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\Theta = (\alpha, \beta)$. The most commonly used distance measure is the Euclidean distance, but any distance, $d_{i,j}$, that follows the triangle inequality, $d_{i,j} \leqslant d_{i,k} + d_{k,j}$, for all i, j, k triples could be used. The resulting log likelihood function of the distance model is

$$\log \mathbb{P}[A|\eta] = \sum_{i \neq j} \{\eta_{i,j} A_{i,j} - \log(1 + \exp^{\eta_{i,j}})\}. \tag{2.17}$$

Similar to the distance model, the projection model is also parametrized using the logistic regression model and is given by

$$\log \frac{p_{i,j}}{1-p_{i,j}} = \alpha + \beta^T \mathbf{c}_{i,j} + \frac{\mathbf{x}_i^T \mathbf{x}_j}{\| \mathbf{x}_j \|_F}. \tag{2.18}$$

The model is based on the assumption, that the probability of observing an edge between two vertices, i and j, depends on the angle that they create in the latent space, that is, two vertices have a higher probability of having an edge if the angle is acute, and a lower probability if the angle is obtuse.

The latent space model successfully represents transitivity through the latent space (X) and can also capture homophily by vertex attributes through covariates (C). For the distance and projection models discussed above, Hoff et al. (2002) do not specify a prior distribution for the latent positions. Nevertheless, by defining a suitable prior such as a Gaussian mixture distribution, it is possible to generate graphs that mimic the clustering property of real-world networks.

2.3.2.1 Latent Position Cluster Model

Handcock et al. (2007) extend Hoff et al. (2002)'s LSM to a *latent position cluster model* to provide for the clustering property by additionally assuming that the latent positions, $\mathbf{x}_i$, are generated from a Gaussian mixture model (GMM) (Banfield and Raftery, 1993).

Thus, in terms of A, the likelihood function of the model is given by Equation 2.15, where each latent position, $\mathbf{x}_i$, in a latent space of dimension d, is assumed to be drawn from a mixture of d variate normal distributions. Using $\phi_d(.|\mu, \Sigma)$ to denote a d-variate normal probability density function (pdf) with mean vector μ and covariance matrix Σ, they assume the following distribution for the latent positions:

$$\mathbf{x}_i \sim \sum_{g=1}^{G} \lambda_g \phi_d(\mathbf{x}_i|\mu_g, \sigma_g^2 I). \tag{2.19}$$

Here, parameter λ_g is the probability that a vertex is drawn from the gth mixture component, $\sum_{g=1}^{G} \lambda_g = 1$, and μ_g and $\sigma_g^2 I$ are the mean vector and the diagonal covariance matrix respectively, of the $d-$variate normal distribution for component g, where I is the $d \times d$ identity matrix. Each mixture component has its own mean vector, μ_g, and represents a cluster of vertices in the graph. The spherical covariance matrix, $\sigma_g^2 I$, implies that the likelihood function is invariant under rotations in the latent space. The authors introduce a fully Bayesian approach using MCMC sampling to simultaneously estimate the latent positions of vertices, cluster memberships, mixture model parameters, and the number of clusters. A major drawback of the MCMC sampling approach is its computational complexity, which is a crucial issue when analysing large graphs.

2.3.2.2 Random Dot Product Graph Model (RDPG)

The random dot product graph model (RDPG) by Nickel (2007) and Young and Scheinerman (2007) is a special case of the latent space models introduced by Hoff et al. (2002). In this model, the probability of an edge between vertices i and j is calculated from the dot product, $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, of their latent positions. In terms of the adjacency matrix, A, for a simple graph, the likelihood function of the model is given by

$$\begin{aligned} \mathbb{P}[A|X] &= \prod_{i \neq j} \mathbb{P}[A_{i,j}|\mathbf{x}_i, \mathbf{x}_j] \\ &= \prod_{i \neq j} \langle \mathbf{x}_i, \mathbf{x}_j \rangle^{A_{i,j}} (1 - \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^{1-A_{i,j}}, \end{aligned} \tag{2.20}$$

subject to the constraint, $0 \leq \langle \mathbf{x}_i, \mathbf{x}_j \rangle \leq 1$. The dot product between two vectors depends on the magnitudes of the vectors and the angle between them. In a social network, where vertices may represent people and edges their friendships, the components of a latent vector may capture social interests of a person whilst the magnitude of the vector reflects how outgoing a person is. Thus, the RDPG model captures the tendency of people with common interests to form relationships. Furthermore, people who are more outgoing are more likely to form relationships (Suwan, 2015).

2.3.3 The Stochastic Block Model (SBM)

The stochastic block model (SBM) (Holland et al., 1983; Wang and Wong, 1987) assigns each vertex in a graph to one of K blocks. A *block* is defined as a group of vertices in the graph that are *stochastically equivalent*[2]. Vertices i and j are stochastically equivalent if $\mathbb{P}[i \text{ connects to } l] = \mathbb{P}[j \text{ connects to } l]$ for every vertex l in the graph.

In the SBM, each vertex, v_i, is assigned to one of K blocks with probability, $\rho = \{\rho_1, \rho_2, \ldots, \rho_K\}$, where ρ_k is the probability of a vertex being assigned to block k, and $\sum_{k=1}^{K} \rho_k = 1$. Let $c_i \in \{1, 2, \ldots, K\}$ denote the block membership of vertex i, then the vector of dimension n, $\mathbf{c} \in \{1, 2, \ldots, K\}^n$, denotes the block memberships of the n vertices in the graph. The probability of an edge is parametrized by $\mathbf{c}$, and by the block probability matrix, $B \in [0, 1]^{K \times K}$, where each element, $B_{s,t}$, gives the probability of an edge between a vertex in block s and a vertex in block t. In terms of the binary adjacency matrix, A, the likelihood for a simple graph under the SBM is given by

$$\mathbb{P}[A|B, \mathbf{c}] = \prod_{i \neq j} (B_{c_i, c_j})^{A_{i,j}} (1 - B_{c_i, c_j})^{1 - A_{i,j}}. \tag{2.21}$$

Thus, conditioned on B and $\mathbf{c}$, each $A_{i,j}$ is distributed as a Bernoulli random variable with parameter B_{c_i, c_j}. Holland et al. (1983) assume $\mathbf{c}$ to be known as a priori and obtain maximum likelihood estimates of B. However, in reality we do not know $\mathbf{c}$ prior to analysis. Numerous approaches have been introduced to estimate $\mathbf{c}$. Some examples include, Bayesian methods (Snijders and Nowicki, 1997), spectral methods (Rohe et al., 2011; Sussman et al., 2012), and modularity maximization (Bickel and Chen, 2009).

Sussman et al. (2012) further extend the SBM to have a RDPG model (Section 2.3.2.2) representation. They redefine the block probability matrix, B, as a function of the latent positions associated with the vertices in the following manner. First, let ν_k be a d-dimensional latent vector associated with block k, where d is assumed to be equal to the rank of B. For $s, t \in \{1, \ldots, K\}$ suppose that $\nu_s \nu_t^T \in [0, 1]$, and let $B_{st} = \nu_s \nu_t^T$. Letting $\nu = (\nu_1, \ldots, \nu_k)$, the SBM can be reparametrized by ν and by $\mathbf{c}$ as

$$\mathbb{P}[A|\nu, \mathbf{c}] = \prod_{i \neq j} (\nu_{c_i} \nu_{c_j}^T)^{A_{i,j}} (1 - \nu_{c_i} \nu_{c_j}^T)^{1 - A_{i,j}}. \tag{2.22}$$

Finally, for $i \in \{1, \ldots, N\}$, by letting $\mathbf{x}_i = \nu_{c_i}$, we have the SBM expressed as a RDPG model.

The SBM given in Equation 2.21 identifies blocks of vertices in simple graphs where the edges are unweighted. However, most of the real-world graphs contain weighted

[2]Stochastic equivalence is also known as structural equivalence.

edges. Discarding the edge weights and representing edges merely as binary variables may give misleading results about the latent block structure. To address this problem, Aicher et al. (2014) proposed the weighted stochastic blockmodel (WSBM). By selecting an alternative probability distribution from the exponential family of distributions, the WSBM generalizes the SBM to model edge weights. For example, in terms of a weighted adjacency matrix with real valued edge weights, $W_{i,j} \in \mathbb{R}$, the likelihood function of the WSBM be written in the form of an exponential family as

$$\mathbb{P}[W|\mathbf{c}, \mu, \sigma^2] = \prod_{i \neq j} \exp\left(W_{i,j}.\frac{\mu_{c_i,c_j}}{\sigma^2_{c_i,c_j}} - W^2_{i,j}.\frac{1}{2\sigma^2_{c_i,c_j}} - 1.\frac{\mu^2_{c_i,c_j}}{\sigma^2_{c_i,c_j}} \right).$$

Instead of just the block probability matrix, B, in Equation 2.21, each edge, $W_{i,j}$, is now richly parametrized by the mean, μ_{c_i,c_j}, and the variance, $\sigma^2_{c_i,c_j}$, for edges connecting block c_i and c_j.

The concept of stochastic equivalence of vertices within blocks in the SBM implies that two vertices in the same block will have identical expected degrees, which is not realistic for real-world graphs. For graphs containing vertices whose expected degrees vary within blocks, the SBM tends to incorrectly partition them into blocks containing high degree and low degree vertices (Zhao et al., 2012). To address this issue, Karrer and Newman (2011) extend the SBM to a degree corrected stochastic block model (DCSBM) by adding a degree parameter, θ, for each vertex in order to allow for a heterogeneous degree distribution within each block.

2.3.3.1 The Degree Corrected Stochastic Block Model

The DCSBM assumes that the number of edges between vertices, i and j, is a Poisson random variable with expected value, $\theta_i B_{c_i,c_j} \theta_j$. The parameter, $B_{c_i c_j}$, denotes the probability of an edge between vertices in blocks c_i and c_j, while the degree information on vertex i is captured by θ_i. In terms of the weighted adjacency matrix, W, of an undirected graph with edge weights corresponding to non-negative Poisson counts, the likelihood function for the DCSBM is given by

$$\mathbb{P}[W|\boldsymbol{\theta}, B, \mathbf{c}] = \prod_{i \neq j} \frac{(\theta_i \theta_j B_{c_i,c_j})^{W_{i,j}}}{W_{i,j}!} \exp(-\theta_i \theta_j B_{c_i,c_j}). \tag{2.23}$$

The model is subject to the constraint that, $\sum_i \theta_i \delta_r(c_i) = 1$ for each $r \in \{1, \ldots, K\}$, where $\delta_r(c_i)$ is 1 if vertex i is in block r and 0 otherwise. This ensures that the degree parameters of all vertices within a block are normalized by summing to one. In this

thesis, we use the DCSBM to generate synthetic graphs for simulation experiments (Section 3.3.2).

2.4 Change Detection in Dynamic Networks

As detailed in Chapter 1, changes in a dynamic network are deviations from usual activities, and change detection is the process of monitoring a time sequence of graphs to detect such deviations. In this section, we summarize existing methods of change detection. Most of the reviewed methods follow a common procedure as follows:

1. Given a time sequence of graphs, determine the smallest unit in the graph (e.g., a vertex, an edge, a subgraph or the entire graph) that needs to be monitored over time to detect the change.
2. At each time instant, extract a good summary measure from the graph that accurately represents the behaviour of the smallest unit selected.
3. Calculate the dissimilarity between the summary measure obtained for the current time instant and a summary measure representing the profile behaviour of the unit during the recent past time instants[3]. This dissimilarity measure is the *change score* for the current time instant.

Based on the unit selected, we divide the change detection methods reviewed into four main categories. They are methods that detect,

1. changes in the overall graph (Section 2.4.1),
2. changes in vertices (Section 2.4.2),
3. changes in edges (Section 2.4.3), and
4. changes in subgraphs (Section 2.4.4). In this category, we separately discuss methods on communities (Section 2.4.4.1), which are a special type of subgraph.

Within each category we also discuss methods that use vertex attributes and edge attributes.

2.4.1 Changes in the Overall Graph

A change in the overall graph occurs when the majority of entities in a network change their usual relationship patterns. For example, Akoglu and Faloutsos (2010) observe a

[3]From Section 1.2, if t is the current time instant, the recent past time instants are the set of w time instants, $\{t-w, \ldots, t-1\}$, prior to it.

mobile communication network over a six months period and discover that most people change their usual communication patterns during the Christian New Year. Changes in the behaviour of most vertices or edges in the graph are considered to be a change in the behaviour of the overall graph. Detecting changes in the overall graph is sometimes referred to as *event detection* (Ranshous et al., 2015).

In a computer network, we can obtain information regarding a set of computers and their interactions over time. Each interaction between a pair of computers contains attribute information such as connection type or number of connections. Idé and Kashima (2004) represent such a computer network as a weighted and undirected graph, where the edge weights are the number of connections between two computers at any given time instant. Each weighted graph is then represented as a weighted adjacency matrix, W. The goal of Idé and Kashima (2004)'s change detection method is to find time instants at which the majority of the edge weights in the graph show significant deviation from the recent past. As it can be computationally inefficient to track each edge over time, the authors employ a spectral decomposition approach for this purpose. From Section 2.2.1, a positive semi-definite symmetric matrix, W, can be decomposed as $W = U\Sigma U^T$, where U is an orthonormal matrix with columns containing the unit eigenvectors of W, and Σ is a diagonal matrix containing the corresponding eigenvalues. Idé and Kashima (2004) extract the principal eigenvector corresponding to the maximum eigenvalue and use it as the representative summary of the graph. The principal eigenvector is a non-negative vector, where larger elements correspond to vertices that have high connectivity or have neighbours with high connectivity in the graph. Hence, Idé and Kashima call this vector the *activity vector*, $\mathbf{u}^t$. This way, the time sequence of graphs can now be represented as a time sequence of activity vectors. At each time instant, t, singular value decomposition (SVD) is performed on the matrix whose columns are the activity vectors from the recent past time instants, and the profile behaviour is represented by the principal left singular vector, $\mathbf{r}^{t-1}$, which Idé and Kashima (2004) call the *typical activity vector*. Finally a change score, z^t, is obtained for each time instant by calculating the dot product between $\mathbf{u}^t$ and $\mathbf{r}^{t-1}$ given by

$$z^t = 1 - < \mathbf{u}^t, \mathbf{r}^{t-1} > . \tag{2.24}$$

As $\mathbf{u}^t$ and $\mathbf{r}^{t-1}$ are unit vectors, z^t corresponds to the cosine of the angle between $\mathbf{u}^t$ and $\mathbf{r}^{t-1}$. Greater deviations from the recent past behaviour correspond to larger angles which give higher change scores. The activity vector at the detected change point is further investigated to detect the vertices that are mostly responsible for the change. Figure 2.1 provides a summary of Idé and Kashima (2004)'s change detection method.

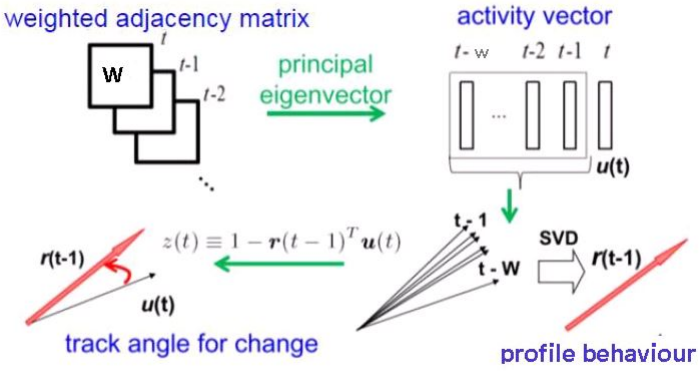
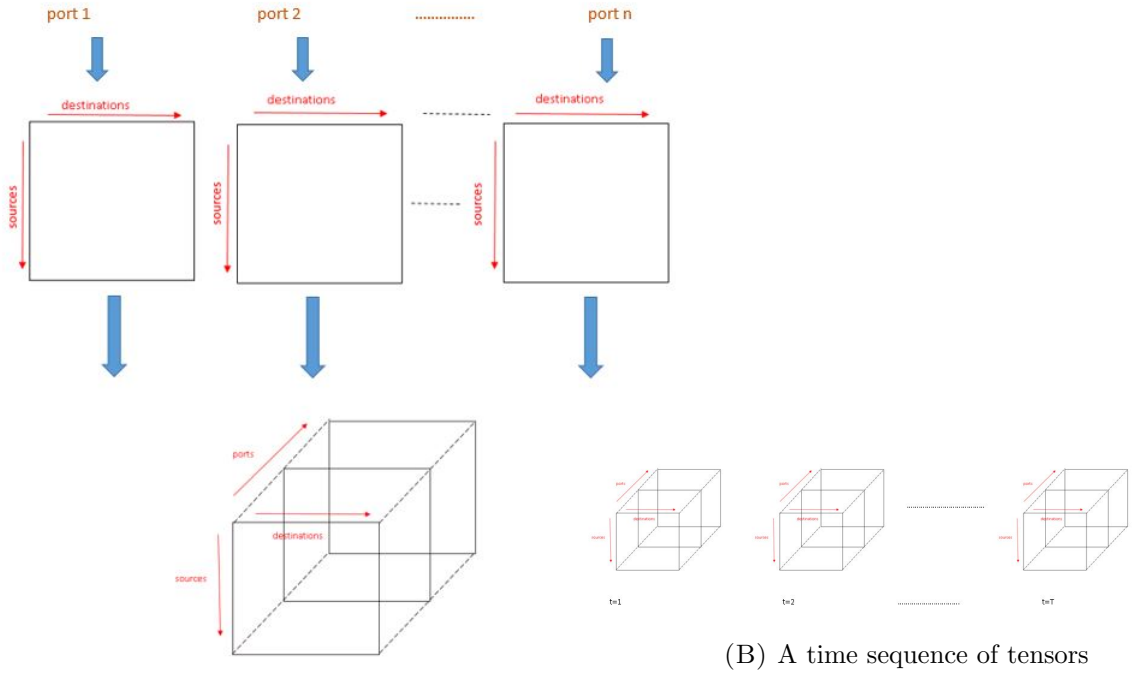


FIGURE 2.1: **Illustration of Idé and Kashima (2004)'s change detection procedure (taken from Akoglu and Faloutsos (2013) and used with permission).** A time sequence of weighted adjacency matrices is first converted into a time sequence of activity vectors. The profile behaviour is obtained using SVD. Finally, the change score is calculated using the dot product between the current activity vector, and corresponding profile vector.

From Section 2.1, a multi-view network is conceptualized as a multi-graph, which is a graph with multiple edge attributes. When a graph has multiple edge attributes, a tensor is preferable to a matrix in order to capture the complex connectivity structure. A tensor is a multidimensional array (Kolda and Bader, 2009)[4]. There are two ways to represent a dynamic multi-view network as a tensor. First, a dynamic multi-view network can be represented as a time sequence of three-dimensional tensors. Here, the first two dimensions of each tensor denote the vertices, and the third dimension denotes the edge attributes. For example, consider a dataset containing port usage in a computer network observed at specific time instants. This dataset may contain the source IP addresses, destination IP addresses, and the port numbers used. Each time instant can then be represented as a tensor, where the first two dimensions represent the sources and destinations, and the third dimension represents the port numbers. The tensor elements are binary, where a 1 indicates a communication between the corresponding source and destination via the corresponding port. Figure 2.2 (A) shows how the sources, destinations, and ports are represented as a three dimensional tensor, and Figure 2.2 (B) shows a dynamic network as a time sequence of three-dimensional tensors.

The second way is to represent the whole multi-view dynamic network as a four dimensional tensor. The first three dimensions constitute the tensor in the first method, while time is added as the fourth dimension. For example, the time sequence of three dimensional tensors in Figure 2.2B can be represented as a single four dimensional tensor with the additional dimension representing the temporal profile of the edges.

[4]Note that a matrix is a special type of tensor, that is of dimension two.



(A) Representing multiple edge attributes as a 3-D tensor

(B) A time sequence of tensors

FIGURE 2.2: **The tensor-based representation of a dynamic multi-view network**. At each time instant, the multiple adjacency matrices corresponding to each edge attribute are stacked along the third dimension of a tensor. The time sequence of multi-graphs is then represented as a time sequence of tensors.

The majority of tensor-based change detection methods use tensor decomposition (further details on tensor decomposition methods are provided in Section 2.5.2.3) to obtain a low-rank approximation, $\tilde{\mathbf{A}}$, of the original tensor, $\mathbf{A}$, at each time instant. The reconstruction error, $e = \| \mathbf{A} - \tilde{\mathbf{A}} \|_F^2$, measures how well the tensor is approximated at that time instant. Sun et al. (2006) decompose each tensor, $\mathbf{A}$, and obtain the reconstruction error, e. If e is 2 standard deviations away from the mean of error values so far, that time instant is characterized as a change point. The tensor-based change detection methods that we find in the literature mainly differ by the technique employed to decompose the tensor. For example, Kolda and Sun (2008) detect change points by doing Memory Efficient Tucker (MET) decomposition. MET addresses the *immediate blow-up problem* which occurs when the input and output tensors exceed the available memory space during computations for large and sparse graphs. Papalexakis et al. (2012), on the other hand, employ the PARAFAC tensor decomposition technique (Harshman, 1972) in their change detection method. For a comprehensive discussion on tensors and tensor decomposition methods, we direct the reader to Kolda and Bader (2009). Furthermore, a collection of tensor-based anomaly detection methods is extensively surveyed in Fanaee-T and Gama (2016).

2.4.2 Changes in Vertices

Vertex-based methods are very useful when we want to track the behaviour of important entities in the network. Akoglu and Faloutsos (2010) optimize the decomposition-based strategies discussed in Idé and Kashima (2004) (Section 2.4.1) to detect changes in vertex behaviour. The inputs to their algorithm are twelve vertex attributes observed at each vertex over time. The whole dataset is first represented as a three-dimensional array, where the three dimensions are the N vertices, F attributes, and $\mathcal{T}$ time instants (Figure 2.3). To detect changes of vertex i, the $\mathcal{T} \times F$ slice of the array corresponding

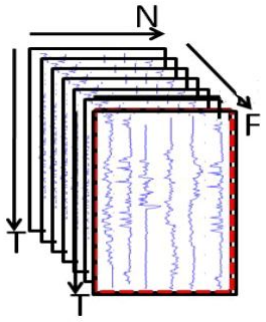


FIGURE 2.3: The representation of the network data as a three dimensional array (taken from Akoglu and Faloutsos (2010) with permission).

to vertex i is first selected. A window of size w is then defined as shown in Figure 2.4. The correlation between each pair of attributes within the window is calculated using

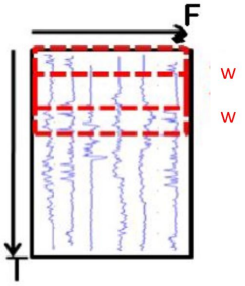


FIGURE 2.4: Illustration of the moving window approach over the $\mathcal{T} \times F$ slice corresponding to vertex i (taken from Akoglu and Faloutsos (2010) with permission).

Pearson's correlation coefficient, resulting in a correlation matrix for all F attributes. By moving the window to cover all $\mathcal{T}$ time instants, a time sequence of correlation matrices is obtained. Let $u^t_{i,f}$ be the value of attribute f for vertex i at time instant t. The $\mathcal{T} \times F$ slice corresponding to vertex i includes the set of time sequences, $\{u^t_{i,f}\}$ for $f \in \{1, \ldots, F\}$ and $t \in \{1, \ldots, \mathcal{T}\}$. Let C^t_i be defined as the matrix calculated at time instant t for vertex i, where each element, $[C^t_i]_{f',f''}$, gives the absolute value of the

correlation between $u^t_{i,f'}$ and $u^t_{i,f''}$ by

$$[B^t_i]_{f',f''} = \frac{1}{w} \sum_{s=t-w+1}^{t} (u^s_{i,f'} - \bar{u}^t_{i,f'})(u^s_{i,f''} - \bar{u}^t_{i,f''}), \tag{2.25}$$

where

$$\bar{u}^t_{i,f'} = \frac{1}{w} \sum_{s=t-w+1}^{t} u^s_{i,f'}, \tag{2.26}$$

and

$$[C^t_i]_{f',f''} = \left| \frac{[B^t_i]_{f',f''}}{\sqrt{[B^t_i]_{f',f'}[B^t_i]_{f'',f''}}} \right|. \tag{2.27}$$

Each element in C_i denotes the strength of the relationship between each pair of attributes. Thus, matrices $C^1_i, \ldots C^{\mathcal{T}}_i$ are analogous to the weighted adjacency matrices analysed in Idé and Kashima (2004) (Section 2.4.1). Akoglu and Faloutsos (2010) apply Idé and Kashima (2004)'s activity vector-based change detection procedure over $C^1_i, \ldots C^{\mathcal{T}}_i$ to detect changes in vertex i's behaviour. However, for a large dynamic network consisting of a large number of entities, it would not be computationally efficient to detect changes in every vertex's behaviour using this method. Hence, this method is more suitable to track a small number of selected vertices that are identified as interesting in advance.

2.4.3 Changes in Edges

These methods aim to detect individual edges or groups of edges that show changes in their behaviour. Some examples of edge-based changes include the appearance of an edge between two vertices in the graph which represents entities that are very unlikely to interact with each other, or a large increase or decrease in the weight of an edge compared to the recent past. For example, in an author conference network (Section 2.1), we may observe an author who usually publishes papers only in conference proceedings related to a particular research area. However, at a certain point in time, he may wish to switch to a new research area and publishes a paper in a related conference proceedings, forming an irregular connection with respect to his past connections in the network (Koutra et al., 2012).

Sricharan and Das (2014) use a method based on *commute time distances* to detect changes involving edges. Given a graph, the commute time distance between vertex i and vertex j is the expected return path length between vertex i and vertex j (Lovász, 1993). A fast and reliable method was introduced by Khoa and Chawla (2012) to calculate commute time distances between vertices in large graphs. Sricharan and Das

(2014) adopt this method to detect edges that undergo changes in a time sequence of graphs with weighted edges. Their change detection method, CAD (commute time-based anomaly detection) measures change in the commute time distance between two vertices in a graph inorder to detect changes in the associated edges. First, the commute time distances between every pair of vertices in a graph are calculated. The change-score, $Z^t_{i,j}$, for an edge between vertex i and vertex j is then given by

$$Z^t_{i,j} = |W^t_{i,j} - W^{t-1}_{i,j}| \times |C^t_{i,j} - C^{t-1}_{i,j}|, \tag{2.28}$$

where W^t is the weighted adjacency matrix and C^t is the commute time distance matrix. A change in the edge weight between vertices, i and j, in turn affects the commute time distances between the neighbours of vertex i and vertex j. Measuring the change using C^t and C^{t-1} alone has been shown to result in many high change scores even for unchanged edges. Thus, Sricharan and Das (2014) include the term, $|W^t_{i,j} - W^{t-1}_{i,j}|$, in Equation 2.28 to address this problem of false detections. According to the experimental results in Sricharan and Das (2014), CAD shows good performance in detecting changing relationships in many real-world networks. However, as CAD can only perform pairwise comparisons between consecutive graphs, it cannot be applied when several graphs from the recent past are required to characterize usual behaviour.

In Section 2.4.1, we discuss how the reconstruction error of a tensor can be employed to detect changes in a dynamic multi-view network. Let $\mathbf{A}^t$ be the tensor at time instant t at which a global structural change has occurred. The search can be further narrowed down to detect edges that have changed their behaviour the most. This can be achieved by examining the elements of the residual tensor, $\mathbf{R}^t = |\mathbf{A}^t - \tilde{\mathbf{A}}^t|$ (Ranshous et al., 2015). The elements in $\mathbf{R}^t$ that show the highest scores correspond to edges that have changed most.

2.4.4 Changes in Subgraphs

In some scenarios, unusual activity spreads throughout a certain neighbourhood in a network and prevails for some time period. For example, in a road network that consists of street intersections connected by road segments, an accident may suddenly increase the traffic density in road segments near to the accident. Such a change is reflected as a change in the behaviour of some vertices and edges in the corresponding graph, where road segments are denoted as edges and street intersections are denoted as vertices. In this case, rather than observing the behaviour of individual vertices or edges, it is more useful to keep track of a *local region* in the graph. A local region is a subgraph formed by a set of vertices that lie in close proximity to each other (Idé et al., 2009), such as, a

set of vertices in a *k-path* (discussed later in the section) (Neil et al., 2013), or a *clique*[5] (Chen et al., 2012).

The concept of a *scan statistic* originated in spatial statistics and image analysis to detect regions of unusual activity (Marchette, 2012). Scan statistics were then adopted in the area of network analysis to detect anomalous regions in a graph. Usually, a subgraph is defined centred at each vertex in the graph and a statistic is calculated. The subgraphs that show unusual values for this statistic are highlighted as anomalous. Priebe et al. (2005) use a scan statistics approach for change detection by detecting a subgraph that shows an unusual increase in interactions with respect to the recent past. The authors apply this method to a time sequence of graphs extracted from the Enron e-mail corpus[6]. Each graph represents the email communications between a group of Enron executives during a week, and contains directed, binary edges. The *kth order neighbourhood* of vertex i is the set of all vertices that lie within a shortest path length of at most k in the graph. The *locality statistic* for vertex i, $\Psi^t_{k,i}$, at time instant t is defined as the size of its k^{th}-order neighbourhood[7]. The maximum of the locality statistics, $\max_i \Psi^t_{k,i}$, is identified as the *scan statistic* at that time instant. By moving a window of length w over the time sequence of scan statistics, a change score is obtained at each time instant. Time instants that give high change scores are detected as change points. For a given change point, t, the vertices corresponding to $\max_i \Psi^t_{k,i}$ are regarded as the vertices responsible for the detected change. When $k = 1$, the subgraph defining $\Psi^t_{k,i}$ in Priebe et al. (2005) resembles a star that is centred at vertex i (Figure 2.5).

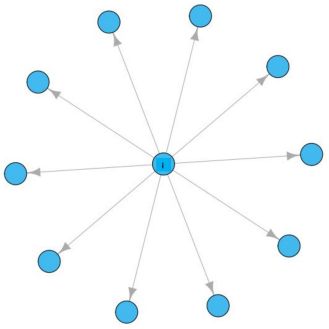


FIGURE 2.5: **The star shaped subgraph in Priebe et al. (2005) for a first order neighbourhood (taken from Neil et al. (2013) with permission)**. The figure depicts an *out star* centred at vertex i.

Computer networks are vulnerable to attacks by intruders. One common type of intruder behaviour is *intruder traversal* (Neil et al., 2013). For example, an intruder sends an email containing a link to a malicious web site to a set of users. When a user clicks on the

[5] A *clique* is a set of vertices, that are pairwise connected to each other.

[6] https://www.cs.cmu.edu/~./enron/

[7] The locality statistic, $\Psi^t_{k,i}$, can also be considered as an attribute attached to a vertex at a given time instant

link, the user's computer gets infected and the intruder gains access to that computer. From this computer the intruder then compromises to another computer in a similar manner. In this way, the intruder traverses the whole computer network, accessing valuable data at each infected computer. Figure 2.6 illustrates the traversal behaviour of an intruder in a network. Neil et al. (2013) introduce a *k-path* shaped subgraph to

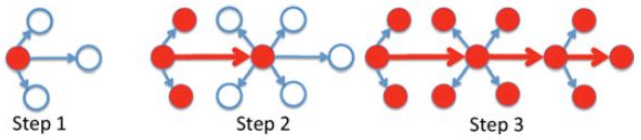


FIGURE 2.6: **The traversal behaviour of an intruder in a computer network (taken from Neil et al. (2013) with permission)**. At step 1 the first computer gets infected. At step 2 the intruder affects the neighbouring computer, and traverses forward. At step 3 a full traversal has occurred. The connections in red in step 3 form a 3-path in the corresponding graph.

detect intruder traversal in a computer network. A k-path is a directed subgraph, where both its size and diameter are equal to k (Kolaczyk, 2009). Neil et al. (2013) obtain scan statistics by fitting a Hidden Markov Model (HMM) on the edges of each k-path in the graph. Similar to the procedure followed in Priebe et al. (2005), k-paths that show different behaviour compared to their recent past are detected using a moving window approach.

2.4.4.1 Changes in Communities

A *community* is a special type of subgraph. There exist several definitions of a community in the network analysis literature. In some publications that use probabilistic model-based approaches to detect community structure in graphs, a community is defined as a set of vertices that are stochastically equivalent. For example, the SBM and its variants discussed in Section 2.3, follow this definition. In other publications, a community is defined as a dense subgraph. Some examples along this line of research include, community detection through spectral clustering (Ng et al., 2001), hierarchical clustering (Newman, 2004b) and maximization of the so called "Newman–Girvan" modularity function (Newman, 2004b). In all the latter mentioned publications, a community consists of a group of vertices that have higher number of edges within the group than between the group and the rest of the graph. For example, in a social network, a community may correspond to a board of directors in a company who have stronger connections among themselves than with others.

Communities in real-world networks are usually stable over time (Koujaku et al., 2013). Thus, deformation in community structure is a good indication of a change occurring in

the network. Chen et al. (2012) define six basic changes that can occur in community structure (Figure 2.7):

1. *Grown community*
   Vertices get added into a smaller community, causing the community to grow into a larger community. The size of the community at time instant t is greater than its size at $t-1$.

2. *Shrunken community*
   Community members leave a community, causing the community to become smaller. The size of the community at time instant t is less than its size at $t-1$.

3. *Merged community*
   At time instant $t-1$, there were two communities. These two communities join together and form a single larger community at time instant t.

4. *Split community*
   One community at time instant $t-1$ divides into two communities at time instant t.

5. *Born community*
   The edges between a set of vertices increase and form a community at time instant t. This scenario uses the same notion as the *form* scenario discussed in Peel and Clauset (2015).

6. *Vanished community*
   A community that prevailed at time instant $t-1$ disappears at time instant t because relationships between members cease to exist. The same change is also referred to as *fragment* in Peel and Clauset (2015).

Koujaku et al. (2015) perform experiments to detect the above-mentioned six types of changes in community structure. Their method employs two moving windows for change score calculation. At each time instant, the first window contains time instants from $t-w+1$ to t and the second window contains time instants from $t+1$ to $t+w$. Let G^- be the graph constructed from the time instants in the first window and let G^+ be the graph constructed from the time instants in the second window. A set of communities, $\mathcal{U}^-$, and another set, $\mathcal{U}^+$, are extracted[8] from G^- and G^+, respectively. A change score, z^t, is then calculated by using the *Variational Information* (VI) (Meilă, 2003) between

[8] Koujaku et al. (2015) explore three algorithms, (i) graph scan (Wang et al., 2008), (ii) extraction method (Zhao et al., 2011), and (iii) $\rho-$dense core (Koujaku et al., 2013), for community extraction.

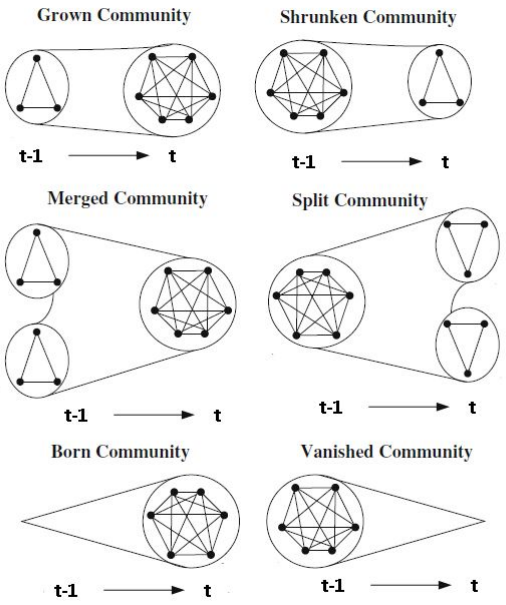


FIGURE 2.7: Possible changes occurring in community structure (taken from Chen et al. (2012) with permission).

$\mathcal{U}^+$ and $\mathcal{U}^-$. This is given by

$$z^t = \text{VI}(\mathcal{U}^+, \mathcal{U}^-) \tag{2.29}$$

$$= H(\mathcal{U}^+) + H(\mathcal{U}^-) - 2I(\mathcal{U}^+, \mathcal{U}^-), \tag{2.30}$$

where H and I are the *entropy* and *mutual entropy* defined as

$$H(\mathcal{U}) = \sum_{V' \in \mathcal{U}} \frac{|V'|}{|V|} \log\left(\frac{|V'|}{|V|}\right), \tag{2.31}$$

$$I(\mathcal{U}, \mathcal{U}') = \sum_{V' \in \mathcal{U}} \sum_{V'' \in \mathcal{U}'} \frac{|V' \cap V''|}{|V|} \log\left(\frac{|V||V' \cap V''|}{|V'||V''|}\right). \tag{2.32}$$

Finally, once a time sequence of change scores is obtained, a threshold is defined and time instants with change scores exceeding this threshold are detected as change points. Figure 2.8 is an illustration of Koujaku et al. (2015)'s change detection procedure.

In Section 2.4.1, we discuss how we can represent a dynamic network with multiple edge attributes as a four-dimensional tensor. Koutra et al. (2012) represent the traffic measurements of a computer network as an $n \times m \times f \times t$ tensor, $\mathbf{A}$, where the four dimensions represent source IP addresses, destination IP addresses, port numbers of the connections and time instants, respectively. They then do PARAFAC decomposition (Harshman, 1970) and obtain factor matrices, $U \in \mathbb{R}^{n \times f'}, V \in \mathbb{R}^{m \times f'}, W \in \mathbb{R}^{f \times f'}$, and

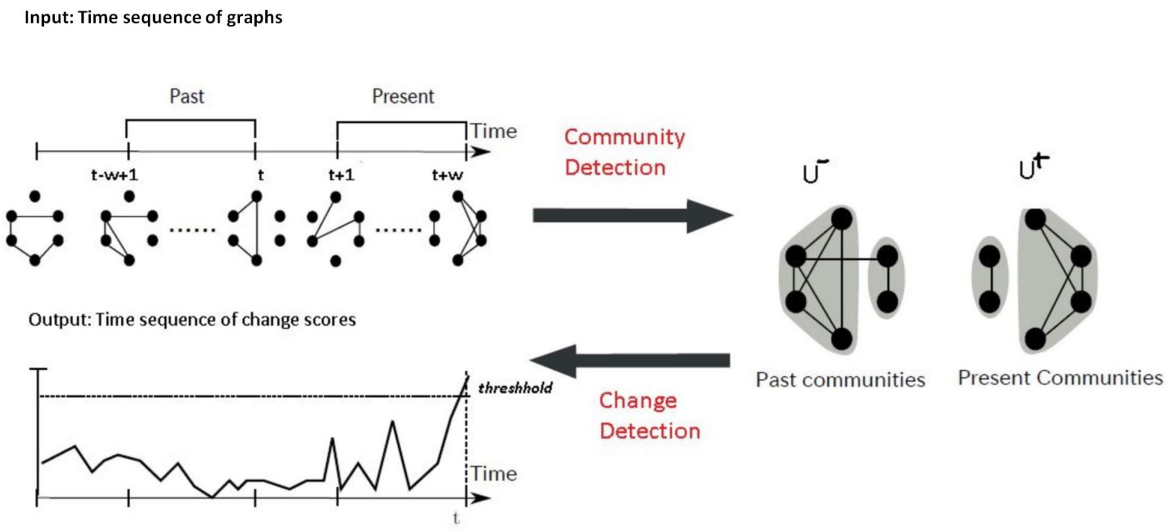


FIGURE 2.8: **Illustration of Koujaku et al. (2015)'s change detection framework with double sliding windows (taken from Koujaku et al. (2015) with permission)**. At each time instant, two sets of communities are detected and a change score is calculated. Finally, a threshold is defined over the time sequence of change scores, and change points are detected.

$X \in \mathbb{R}^{t \times f'}$, where $f' < \min\{n, m, f, t\}$. Each column of U represents a community of sources (source IP addresses), while each column of V represents a community of destinations (destination IP addresses). The columns of W correspond to the port numbers of the connections established between communities represented by the respective columns in U and V, while the columns of X give temporal profiles of these communities. Koutra et al. select the f' most important columns of these factor matrices, track them inorder to investigate changes in user behaviour and to detect intruder attacks in the computer network. For example, Figure 2.9 shows a plot of the same columns taken from each of the factor matrices, U,V, W and X. We observe suspicious behaviour on port 1544. The connection on port 1544 by a particular source IP address to a particular destination IP address is established periodically with perfectly evenly spaced spikes of activity. This type of behaviour is called a *bot attack*. Intruders in computer networks often exhibit this type of behaviour (Koutra et al., 2012). In Figure 2.10, from another common column taken from each of the four factor matrices, we observe an overwhelming burst of traffic concentrated in a certain time interval on port 80 by a particular IP address connected to several destination IP addresses. This type of behaviour resembles a human browsing several websites during a given time interval. Koutra et al. (2012)'s tensor-based change detection method is called *tensor-splat*. Tensor-splat is used for detecting changes in many real-world networks. For example, the DBLP dataset[9] consists

[9] http://dblp.uni-trier.de/xml/

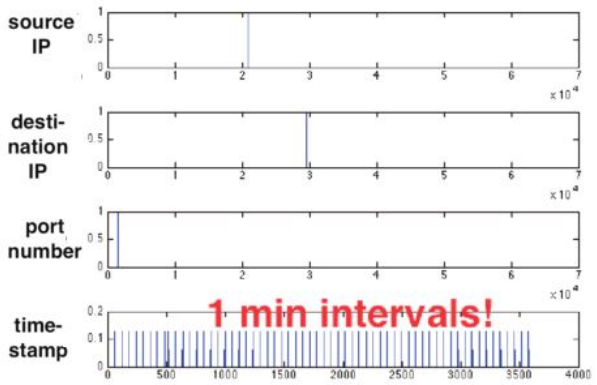


FIGURE 2.9: **Bot-attack like behaviour (taken from Koutra et al. (2012) with permission)**. A connection is established at port 1544 with evenly spaced spikes of 60 seconds.

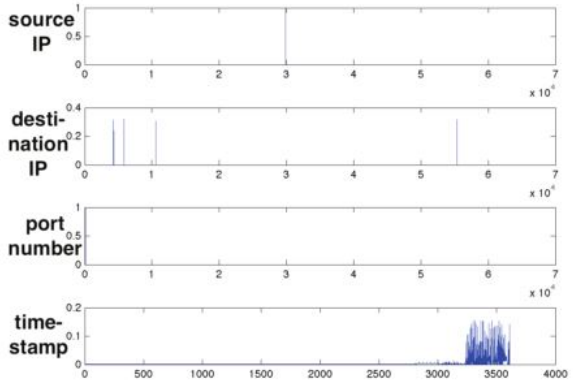


FIGURE 2.10: **Traffic caused by human activity (taken from Koutra et al. (2012) with permission)**. The connection made at port 80 shows high activity concentrated at a short time period. This resembles a human browsing web pages at the indicated destination IP addresses.

of a large collection of papers that appeared at different conferences written by a number of authors over several years. Koutra et al. (2012) build a tensor with dimensions, $418000 \times 3.5000 \times 49$, with the three dimensions representing authors, conferences, and years, respectively. When the tensor is decomposed using PARAFAC as described earlier, the communities indicated by the factor matrix representing authors, signify groups of authors who come from similar research areas. Using tensor-splat, Koutra et al. detect *bridge authors* who gradually changed their research area over time.

A majority of the change detection methods discussed in Section 2.4 calculate change scores by extracting a representative summary from each graph in a time sequence of graphs representing a dynamic network. As real-world graphs can consist of millions of vertices and edges that are often rich with attributes, computationally efficient methods

are needed for studying their structure. Spectral methods have garnered highest popularity in statistics and machine learning, especially with respect to applications handling huge and complex datasets (Belabbas and Wolfe, 2009). In this thesis, we use spectral methods to extract a representative summary from the graph at each time instant. In the next section, we provide a comprehensive discussion on spectral embedding methods.

2.5 Spectral Embedding Methods for Network Analysis

Recall from Section 2.2, an undirected, weighted graph with n vertices can be represented by a weighted adjacency matrix, W, of dimension $n \times n$. Each element, $W_{i,j}$, corresponds to the weight of the edge between vertex i and vertex j, which quantifies the strength of the connection between them. The weighted adjacency matrix, W, itself is often not helpful in gaining insights into the underlying graph structure for several reasons.

1. Data collected in real-world are usually noisy due to issues such as measurement errors or respondent's bias during the data collection process. Thus, there may be missing values, outliers or redundancies in W, that lead to poor results if not addressed prior to analysis.

2. The networks that we encounter are usually large ($n > 1000$) and sparse (Section 2.2). Thus, W is a large matrix where most of its elements are zeros. It is difficult to gain an understanding of the connectivity structure of the underlying network using such a representation.

3. Real-world graphs are most likely to be highly clustered (Watts and Strogatz, 1998). Clusters may be identified by using a suitable measure to calculate the closeness (similarity) between vertices in the graph. For example, the shortest path length (Section 2.2), is one such measure of closeness. However, for large graphs, it is computationally expensive to calculate the shortest path length between every pair of vertices. One might attempt to cluster vertices by using a measure such as Euclidean distance over the rows of W, by considering each row, $W_{i,.} \in \mathbb{R}^n$, as a mapping of a vertex i in the graph to a point in geometric space. However, when W is large and sparse, the Euclidean distances fail to reflect the actual closeness between vertices in the graph (Skillicorn, 2007a).

Using spectral methods, we obtain a low dimensional representation that is simpler than W and further enhances the implicit connectivity structure of the graph. Specifically, each pair of vertices, i and j, in the original n dimensional space is mapped onto a pair

of points, $\mathbf{x}_i$ and $\mathbf{x}_j$, each of dimension $d \ll n$ and referred to as the embeddings[10] of the vertices.

The embedded points in the low dimensional Euclidean space preserve the closeness between vertices in the graph and provide a compact and reliable representation of the underlying complex relationships between them. Below we provide a few examples of different network analysis tasks where spectral embedding methods are employed.

1. Obtaining a latent space representation:
   In Section 2.3.2, we discussed latent space models (LSMs) that associate a latent d-dimensional vector with each vertex. For example, in Hoff et al. (2002)'s distance model, the probability of an edge between vertices, i and j, is based on the distance between the two latent vectors associated to them. Given a large graph, estimating latent vectors in LSMs is a computationally challenging problem. Sussman et al. (2012) introduce a method called *Adjacency Spectral Graph Embedding* to consistently estimate the latent vectors of a random dot product graph[11] represented by the binary adjacency matrix, $A \in [0,1]^{n \times n}$. A more theoretical explanation of this method can be found in Sussman et al. (2014).

2. Spectral clustering:
   A cluster is a group of vertices that have greater similarity amongst themselves than with the rest of the vertices in the graph. *Spectral clustering* applies a clustering algorithm such as k-means on the embeddings to find clusters in the graph. This is also known as *community detection*. Rohe et al. (2011) and Qin and Rohe (2013) cluster the embeddings from the Laplacian matrix (Section 2.5.1.1) to consistently estimate the communities in the SBM graph and the DCSBM graph, respectively. A comprehensive review of several other spectral clustering algorithms can be found in Von Luxburg (2007).

   - Sub-structure discovery:
     Substructures are clusters in the graph that are different from others. For example, in a social network, substructures may correspond to some suspicious activity occurring in the network such as collaborations formed by drug dealers or terrorist organizations. Skillicorn (2007b) discusses several methods of employing spectral embeddings to detect substructures.

3. Measuring centrality:
   The measure of an entity's importance in the network is given by the centrality (see

[10] The term *embedding* in this section means that a vertex in the graph is mapped to a point in a finite dimensional Euclidean space, and is different from the usual graph theoretic terminology.
[11] From Section 2.3.2.2, the random dot product graph model is a special type of a LSM.

Section 2.2) of the corresponding vertex in the graph. In Section 2.4.1, we discussed how the principal eigenvector of a weighted adjacency matrix of an undirected graph defines eigenvector centrality of the vertices. The elements of the principal eigenvector can also be considered to be embedded points of vertices, with the dimension being one. Idé and Kashima (2004) track this principal eigenvector over time to detect changes in the centralities of vertices.

The spectral embedding approach used in each of these situations may be different from each other. For example, in multi dimensional scaling (MDS), the distance matrix, $P \in \mathbb{R}^{n \times n}$, (where each element, $P_{i,j}$, denotes the distance between vertices i and j) is used to obtain an embedding of the graph (Borg and Groenen, 2005). Meanwhile, in spectral clustering, the similarity matrix is used to obtain an embedding of the graph (Von Luxburg, 2007). In the following sections, we describe our optimal procedure of obtaining a spectral embedding from an undirected graph. We divide our discussion into two main sections: (i) matrix-based approaches applicable for a single-view network (Section 2.5.1), and (ii) tensor-based approaches for a multi-view network (Section 2.5.2).

2.5.1 Spectral Embedding Methods for Graphs Represented by Matrices

The graph Laplacian (Section 2.5.1.1) and its variants (Sections 2.5.1.2 and 2.5.1.3) play a major role in our spectral embedding procedure. We briefly review some properties of these matrices that are relevant for this thesis. A more thorough discussion can be found in Chung (1997).

2.5.1.1 The Laplacian Matrix

Define D to be the diagonal *degree matrix* with elements, $D_{i,i} = d_i$ (for all $i \in V$), along the diagonal. Here,

$$d_i = \sum_{j=1}^{n} W_{i,j}. \tag{2.33}$$

The Laplacian matrix is defined as

$$L = D - W. \tag{2.34}$$

2.5.1.1.1 Properties of L

1. For every vector, $\mathbf{x} \in \mathbb{R}^n$, it can be shown that

$$\mathbf{x}^T L \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{n} W_{i,j}(x_i - x_j)^2. \tag{2.35}$$

2. The smallest eigenvalue of L is zero with eigenvector, $\mathbf{1}$, which is a constant vector of all ones.

3. From the symmetry of W and D, L is symmetric. From Property 1, L is positive semi-definite (Section 2.2.1). The symmetric positive semi-definite matrix L has n non-negative real valued eigenvalues $\lambda_1 \leqslant \ldots \leqslant \lambda_n$, with $\lambda_1 = 0$.

4. For an undirected graph (weighted or unweighted), the multiplicity of eigenvalue 0 gives the number of connected components in the graph.

For more details and proofs see Von Luxburg (2007, Section 3). There are two variants of L, they are the random walk Laplacian (Section 2.5.1.2), and the symmetric Laplacian (Section 2.5.1.3).

2.5.1.2 The Random Walk Laplacian

The random walk Laplacian is defined as

$$L_{rw} = D^{-1}L. \tag{2.36}$$

2.5.1.2.1 Properties of L_{rw}

1. λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$ if and only if λ and $\mathbf{x}$ solve the generalized eigenvalue problem, $L\mathbf{x} = \lambda D\mathbf{x}$.

2. Let W_{rw} be the *random walk weighted adjacency matrix*, given by $W_{rw} = D^{-1}W$. λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$ if and only if $(1 - \lambda)$ is an eigenvalue of W_{rw} with eigenvector $\mathbf{x}$. Thus, the eigenvectors of L_{rw} and W_{rw} are the same.

For more details and proofs we refer the reader to Appendix A.

2.5.1.3 The Symmetric Laplacian

The symmetric Laplacian is defined as

$$L_{sym} = D^{-1/2} L D^{-1/2}. \tag{2.37}$$

2.5.1.3.1 Properties of L_{sym}

1. For every vector $\mathbf{x}$ of n elements, we have

$$\mathbf{x}^T L_{sym} \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{n} W_{i,j} (\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}})^2.$$

2. Let W_{sym} be the *degree normalized weighted adjacency matrix* defined as $W_{sym} = D^{-1/2} W D^{-1/2}$. λ is an eigenvalue of L_{sym} with eigenvector $\mathbf{x}$ if and only if $(1 - \lambda)$ is an eigenvalue of W_{sym} with eigenvector $\mathbf{x}$. Thus, the eigenvectors of L_{sym} and W_{sym} are the same.

2.5.1.3.2 Common Properties of L_{rw} and L_{sym}

1. If λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$, then λ is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbf{x}$.

2. Zero is an eigenvalue of L_{rw} with eigenvector, $\mathbf{1}$, and zero is an eigenvalue of L_{sym} with eigenvector, $D^{1/2}\mathbf{1}$.

3. L_{sym} and L_{rw} are positive semi-definite matrices and have n non-negative real valued eigenvalues, $\lambda_1 \leqslant \ldots \leqslant \lambda_n$, where $\lambda_1 = 0$.

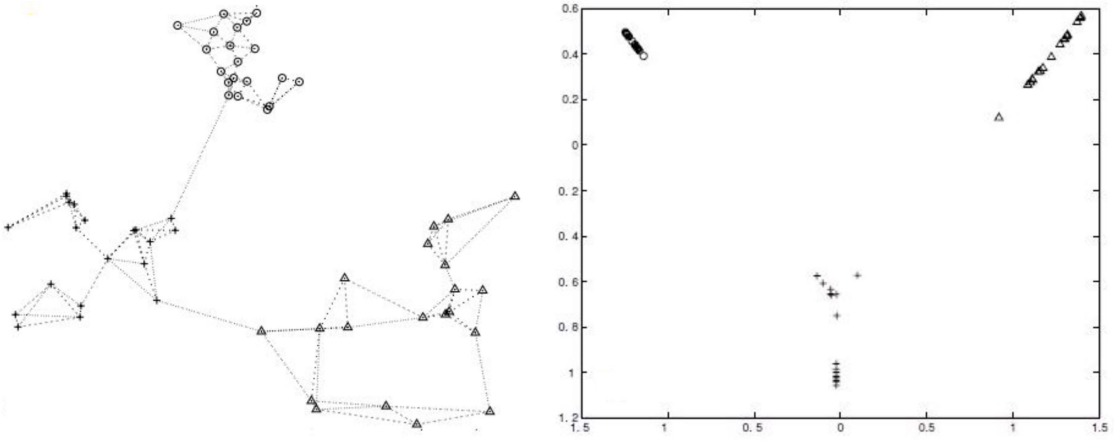
4. For an undirected graph (weighted or unweighted), the multiplicity of eigenvalue 0 of both L_{sym} and L_{rw}, equals the number of connected components in the graph.

For detailed proofs we refer the reader to Von Luxburg (2007, Section 3).

The matrices A, W, L, L_{rw}, W_{rw}, L_{sym}, and W_{sym} can provide different representations of the connectivity structure of a graph. Hence, these matrices are commonly referred to as the *representation matrices* of a graph (Skillicorn, 2007a).

2.5.1.4 Obtaining an Optimal Graph Embedding

Our goal for spectral embedding is to obtain a low dimensional representation of vertices which maintains their edge-based closeness in the graph. For example, in Figure 2.11, we give a simple illustration of an embedding of a graph. Figure 2.11 (A) shows a small graph where the length of each edge is drawn proportionally to the closeness between the corresponding pair of vertices. We can observe three clusters of vertices in this graph. Figure 2.11 (B), gives the two-dimensional embedding, where each vertex is represented as a point in a two dimensional Euclidean space. We can see how the edge-based closeness of vertices in the graph in Figure 2.11 (A) is maintained by the embedded points in Figure 2.11 (B). This characteristic is the *locality preserving property* of the embedded points.



(A) Original graph.

(B) Two-dimensional embedding.

FIGURE 2.11: **Illustration of an embedding of a network (taken from Saerens et al. (2004) with permission)**. The two-dimensional embedding preserves the edge-based closeness in the original graph.

When the embeddings maintain the closeness of the vertices in the original graph, they can in turn reveal the clusters of vertices in the graph. Belkin and Niyogi (2003) show how the objective function optimized for obtaining a low dimensional embedding of a graph which preserves closeness yields the same solution to the spectral clustering objective function proposed by Ng et al. (2001). Below, we briefly review Ng et al. (2001)'s spectral clustering framework and show how it leads to an optimal embedding. Our discussion is based on Von Luxburg (2007, Section 5.3).

Let $G = (V, E)$ be an undirected, weighted graph with vertex set, $V = \{v_1, v_2, \ldots v_n\}$[12], and edge set, E. Let W be the $n \times n$ weighted adjacency matrix of G, where each element, $W_{i,j}$, carries a non-negative weight, $W_{i,j} \geq 0$. Each $W_{i,j}$ denotes the similarity

[12]Recall that v_i denotes vertex i.

or closeness between vertices i and j in G. It is assumed that G is connected. Let B and $\bar{B}$ be two subsets of vertices of G such that $B \neq \varnothing, \bar{B} \neq \varnothing, B \cap \bar{B} = \varnothing$ and $B \cup \bar{B} = V$. The *cut* of B and $\bar{B}$ is the total weight of the edges connecting the vertices in B and $\bar{B}$:

$$\text{cut}(B, \bar{B}) = \sum_{i \in B, j \in \bar{B}}^{n} W_{i,j}. \tag{2.38}$$

The vertices in G can be partitioned into $k \leq n$ subsets, $B_1, B_2, \ldots B_k$, by minimizing the *Ncut* objective function (Shi and Malik, 2000),

$$\text{Ncut}(B_1, B_2, \ldots B_K) \quad = \quad \sum_{k=1}^{K} \frac{\text{cut}(B_k, \bar{B}_k)}{\text{vol}(B_k)}, \tag{2.39}$$

where $\text{vol}(B_k) = \sum_{i \in B_k} d_i$ measures the size of B_k. Unfortunately, the Ncut problem is NP hard (see Wagner and Wagner (1993) for a detailed discussion). Hence, an approximate solution is found by relaxing the Ncut optimization function as follows.

For the case $k = 2$, let us define the *cluster indicator vector* $\mathbf{f}$ by

$$f_i = \begin{cases} \sqrt{\frac{\text{vol}(\bar{B})}{\text{vol}(B)}} & \text{if } v_i \in B, \\ -\sqrt{\frac{\text{vol}(B)}{\text{vol}(\bar{B})}} & \text{if } v_i \in \bar{B}. \end{cases} \tag{2.40}$$

From Equation 2.35, we know that $\mathbf{f}^T L \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} W_{i,j}(f_i - f_j)^2$. It can be shown that

$$\mathbf{f}^T L \mathbf{f} = \text{vol}(V)\text{Ncut}(B, \bar{B}), \tag{2.41}$$

$$(D\mathbf{f})^T \mathbf{1} = 0, \tag{2.42}$$

and

$$\mathbf{f}^T D \mathbf{f} = \text{vol}(V). \tag{2.43}$$

Thus, the Ncut problem in Equation 2.39 with respect to $k = 2$ can be rewritten as

$$\min \frac{\mathbf{f}^T L \mathbf{f}}{\mathbf{f}^T D \mathbf{f}}, \tag{2.44}$$

subject to constraints in Equations 2.40 and 2.42. The optimization is still NP hard as condition 2.40 allows $\mathbf{f}$ to take only two values. This discreteness condition is relaxed by allowing f_i to take arbitrary values in $\mathbb{R}$ by substituting $\mathbf{g} = D^{1/2}\mathbf{f}$. After substitution, the relaxed Ncut optimization formulation can be stated as

$$\min \frac{\mathbf{g}^T D^{-1/2} L D^{-1/2} \mathbf{g}}{\mathbf{g}^T \mathbf{g}}, \tag{2.45}$$

subject to constraints $\mathbf{g} \perp D^{1/2}\mathbf{1}$, where $\perp$ denotes orthogonality, and $\| \mathbf{g} \|_F = \text{vol}(V)$. From Section 2.5.1.3, $D^{-1/2}LD^{-1/2} = L_{sym}$. Thus, Equation 2.45 can be written as

$$\min \frac{\mathbf{g}^T L_{sym}\mathbf{g}}{\mathbf{g}^T\mathbf{g}}. \tag{2.46}$$

Equation 2.46 is in the form of Rayleigh-Ritz theorem. From Section 2.5.1.3.2, $D^{1/2}\mathbf{1}$ corresponds to the first eigenvector of L_{sym}, and $\text{vol}(V)$ is a constant. Thus, the solution, $\mathbf{g}$, is given by the second eigenvector corresponding to the second smallest eigenvalue of L_{sym}. We can re-transform the relaxed solution vector, $\mathbf{g}$, into the discrete indicator vector, $\mathbf{f} = D^{-1/2}\mathbf{g}$. From Section 2.5.1.3.2, we see that $\mathbf{f}$ is the second eigenvector of L_{rw}. Most spectral clustering algorithms cluster the coordinates of $\mathbf{g}$ or $\mathbf{f}$ into two groups to optimally partition the vertices in the graph. Usually, a non-parametric algorithm such as k-means is run on the embedded points for this purpose.

For the case $K > 2$, we define cluster indicator vectors $\mathbf{h}_k = (h_{1,k}, \ldots, h_{n,k})^T$ by

$$h_{i,k} = \begin{cases} \sqrt{\frac{1}{\text{vol}(B_k)}} & \text{if } v_i \in B_k, \\ 0 & \text{otherwise}, \end{cases} \tag{2.47}$$

for $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, K\}$. Taking H as the matrix containing these k cluster indicator vectors as columns, it can be shown that $H^T H = I$, $h_k^T D h_k = 1$, and $h_k^T L h_k = \frac{\text{cut}(B_k, \bar{B}_k)}{\text{vol}(B_k)}$. Combining these facts,

$$\begin{aligned} \text{Ncut}(B_1, B_2, \ldots B_K) &= \sum_{k=1}^{K} h_k^T L h_k \\ &= \sum_{k=1}^{K} [H^T L H]_{k,k} \\ &= \text{trace}(H^T L H). \end{aligned}$$

Hence, the Ncut optimization problem in Equation 2.39 with respect to $k > 2$ can be rewritten as

$$\min \text{trace}(H^T L H)$$

subject to constraints, $H^T D H = I$ and Equation 2.47. Similar to Equation 2.45, we relax the discreteness condition by substituting $U = D^{1/2}H$, and obtain the relaxed optimization formulation,

$$\min_{U \in \mathbb{R}^{n \times k}} \text{trace}(U^T D^{-1/2} L D^{-1/2} U), \tag{2.48}$$

or equivalently,

$$\min_{U \in \mathbb{R}^{n \times k}} \text{trace}(U^T L_{sym} U), \tag{2.49}$$

subject to $U^T U = I$. From the Rayleigh-Ritz theorem, the solution to the trace minimization problem in Equation 2.49 is given by the first k eigenvectors of L_{sym}. Substituting $H = D^{1/2}U$, we see that H is the matrix where columns are the set of first k eigenvectors of L_{rw} (from Section 2.5.1.3.2, the eigenvectors of L_{rw} and L_{sym} only differ by a multiplicative constant $D^{1/2}$). Similar to the $k = 2$ case, a standard spectral clustering algorithm clusters the rows of either H or U to find optimal partitions of vertices in the graph.

From the discussion in Section 2.5.1.4, it is clear that the first k eigenvectors of L_{sym}[13] define the optimal embedding that preserves the closeness of vertices in the graph, and at the same time, emphasizes its cluster structure. Recall from Section 2.5.1.3, that W_{sym} and L_{sym} have the same eigenvectors. Hence, as an alternative, the eigenvectors of W_{sym} can be used to obtain an optimal embedding of the graph. Considering the relationship between L_{sym} and W_{sym}, the optimization function 2.49 can be equivalently formulated as

$$\max_{U \in \mathbb{R}^{n \times k}} \text{trace}(U^T W_{sym} U), \tag{2.50}$$

subject to $U^T U = I$. According to Liu et al. (2013), due to the positive semi-definite property of W_{sym}, the solution U can also be found by solving the following Frobenius norm optimization function

$$\max_{U \in \mathbb{R}^{n \times k}} \| U^T W_{sym} U \|_F^2, \tag{2.51}$$

subject to $U^T U = I$.

2.5.1.5 The Impact of Regularization on Spectral Embedding

The majority of the real-world graphs are sparse and contain vertices with heterogeneous degrees (Sengupta and Chen, 2015). In these graphs, due to the irregularity in the distribution of node degrees, embeddings obtained from L_{sym} or W_{sym} (as discussed in Section 2.5.1.4), do not perform well in representing the true underlying connectivity structure (Joseph and Yu, 2013). In order to address this problem, Amini et al. (2013) propose adjusting the irregularity in the degree distribution by adding a constant of order $\frac{1}{n}$ to each element of the adjacency matrix of the graph. In this thesis, the term *regularization* is used to address Amini et al. (2013)'s method. Below, we briefly outline

[13] One can select either L_{rw} or L_{sym} to obtain an embedding of the graph. From Section 2.5.1.3.2, we know that the embeddings from the two matrices differ only up to a multiplicative constant $D^{1/2}$. However, as only L_{sym} is symmetric, we prefer to use it for the work of this thesis.

the effect of regularization on spectral embedding results. The following discussion is mainly based on Le et al. (2015) and Le and Vershynin (2015).

Let A be the $n \times n$ binary, symmetric adjacency matrix of an undirected graph, G, on the vertex set, $V = \{1, 2, \ldots, n\}$, and edge set, E, where E contains edge, e_{ij}, if vertices, i and j are connected. Let D be the diagonal matrix where elements, $D_{i,i}$, along the diagonal gives the degree of each vertex. Here, $D_{i,i} = d_i$, for all $i \in \{1, \ldots, n\}$ and

$$d_i = \sum_{j=1}^{n} A_{i,j}. \tag{2.52}$$

Then the symmetric Laplacian, L_{sym}, is

$$L_{sym} = D^{-1/2} L D^{-1/2},$$

where $L = D - A$. In Section 2.3, we discussed several random graph models that define a probability distribution on the observed edges in G. Different random graph models assume different probability distributions that define different connectivity structures in the graph. In order to generalize to all types of edge distributions, it is assumed that the edges of the graph are each included independently with different probabilities. Using our usual notation, the likelihood function in terms of A is given by

$$\mathbb{P}[A_{i,j}|P_{i,j}] = \prod_{i \neq j} P_{i,j}^{A_{i,j}} (1 - P_{i,j})^{1 - A_{i,j}},$$

for all $i, j \in \{1, \ldots, n\}$. The symmetric matrix, $P \in [0, 1]^{n \times n}$, is the probability matrix, where each element, $P_{i,j}$, denotes the probability of an edge between vertices, i and j. This model is also called the *independent edge random graph model* in Sussman (2014).

A random graph, G, *concentrates* if its representation matrix is close to its expected value (Le and Vershynin, 2015). For example, if the representation matrix is the adjacency matrix, A, then its expectation is the population adjacency matrix, $\mathcal{A} = E(A)$, where each element, $\mathcal{A}_{i,j} = P_{i,j}$, gives the probability of the presence of edge, $e_{i,j}$. Define $\mathcal{X} \in \mathbb{R}^{n \times d}$ to be an embedding obtained from $\mathcal{A}$. If we are able to obtain $\mathcal{X}$ from $\mathcal{A}$, $\mathcal{X}$ can be used to accurately estimate P[14]. However, $\mathcal{A}$ is unobserved. What we do observe is A. Thus, we obtain an embedding, $X \in \mathbb{R}^{n \times d}$, from A, that converges in the appropriate sense to $\mathcal{X}$ in order to perform accurate inference for network analysis (Rohe et al., 2011). The convergence of X to $\mathcal{X}$ is assured by random graph concentration theorems (Oliveira, 2009).

First, let us formally define *matrix concentration* in terms of L_{sym}.

[14] For example, in the RDPG representation of the SBM discussed in Section 2.3.3, the block probability matrix of the SBM is re-parametrized by the latent vectors associated with the vertices.

Definition 2.1 (Matrix Concentration)**.** Let G be an independent edge random graph on the set of vertices, $V \in \{1, \ldots, n\}$, where each potential edge, $e_{i,j}$, appears with probability, $P_{i,j}$. To assess concentration, we check if

$$\| L_{sym} - \mathcal{L}_{sym} \|_2 \ll 1, \tag{2.53}$$

occurs with high probability for $\mathcal{L}_{sym} = \mathcal{D}^{(-1/2)}(\mathcal{D} - \mathcal{A})\mathcal{D}^{(-1/2)}$, where $\mathcal{A} = E(A) = P$, and $\mathcal{D}$ is the diagonal degree matrix obtained from $\mathcal{A}$. Here, $\| \, . \, \|_2$ denotes the L_2 norm (Equation 2.12).

Oliveira (2009) shows that an independent edge random graph satisfies

$$\| L_{sym} - \mathcal{L}_{sym} \|_2 = \mathcal{O}\left(\sqrt{\frac{\log n}{\hat{d}}}\right) \tag{2.54}$$

with high probability, where $\hat{d}$ is the minimum expected vertex degree of the graph given by

$$\hat{d} = \min_i \sum_j \mathcal{A}_{i,j}. \tag{2.55}$$

In dense graphs, the expected vertex degrees grow at least as fast as $\log n$, so $\hat{d} \gg \log n$. Thus, the condition, $\| L_{sym} - \mathcal{L}_{sym} \|_2 \ll 1$, in Definition 2.1 is satisfied and concentration is assured. Unfortunately, sparse random graphs contain vertices with very low degrees. The minimum expected vertex degree in sparse graphs does not scale as a multiple of $\log n$. Hence, the condition $\hat{d} \gg \log n$ is not satisfied. Thus, $\| L_{sym} - \mathcal{L}_{sym} \|_2$ is much greater than 1 for sparse random graphs, causing failure in the concentration of L_{sym}.

Thus we understand, that the failure in concentration occurs in sparse graphs due to the irregularity in the degree distribution. Amini et al. (2013) propose a method to *regularize* the degree distribution to address this problem. They suggest adding a small constant to each entry in the adjacency matrix. For a regularizer $\tau \geq 0$, the regularized adjacency matrix is

$$A_\tau = A + \tau \mathbf{1}\mathbf{1}^T, \tag{2.56}$$

where $\mathbf{1}$ is an n-dimensional column vector containing all ones. This simple modification adds *weak edges* between all vertices. After applying the above regularization step to the original adjacency matrix, the new representation matrix which is called the *regularized symmetric Laplacian* L_τ is

$$L_\tau = I - D_\tau^{-1/2} A_\tau D_\tau^{-1/2}, \tag{2.57}$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix, and D_τ is the $n \times n$ diagonal matrix with the degrees obtained from A_τ along the diagonal.

By adapting Oliveira (2009)'s results, Le and Vershynin (2015) theoretically show how regularization brings about the concentration of L_{sym}. They show that the regularized symmetric Laplacian, L_τ, of an independent edge random graph satisfies

$$\| L_\tau - \mathcal{L}_\tau \|_2 = \mathcal{O}\left(\sqrt{\frac{1}{\tilde{d}}}\right),$$

where $\tilde{d} = \max_{i,j} nP_{i,j}$ and $1 \ll n\tau \ll \tilde{d}$. By choosing the regularization parameter τ that satisfies the condition, $1 \ll n\tau \ll \tilde{d}$, Le et al. (2015) show that regularization not only confirms that $\| L_\tau - \mathcal{L}_\tau \|_2 \ll 1$, but it also forces L_τ to be near to $\mathcal{L}_{sym}$ ($\| L_\tau - \mathcal{L}_{sym} \|_2 \ll 1$). This shows that regularization not only confirms concentration, but also preserves the original model. Le et al. (2015) further discuss how this result can be easily generalized to different types of graphs such as directed graphs, bipartite graphs, and weighted graphs.

Joseph and Yu (2013) provide a thorough theoretical investigation on how spectral clustering accuracy is influenced by Amini et al. (2013)'s regularization method on a sparse graph generated from the SBM. They show that using a suitable regularization parameter, τ, an embedding from L_τ is assured to converge, providing a more accurate identification of the clusters in the graph. Recall from Section 2.5.1, that spectral clustering and spectral embedding are closely related to each other. Hence, the convergence of embeddings that applies to spectral clustering, similarly applies to spectral embedding. Thus, combining Le and Vershynin (2015)'s and Joseph and Yu (2013)'s results, we conclude that regularization improves spectral embedding results from L_{sym}.

An alternative method to regularize the degree distribution of a graph was proposed by Qin and Rohe (2013). This form of regularization acts directly upon the degree matrix instead of the adjacency matrix. The regularized degree matrix is given by

$$D_\tau = D + \tau I, \tag{2.58}$$

and the corresponding regularized symmetric Laplacian matrix is

$$\bar{L}_\tau = I - D_\tau^{-1/2} A D_\tau^{-1/2}. \tag{2.59}$$

It is important to note that only the regularization proposed by Amini et al. (2013) (Equation 2.56) ensures the resulting graph is connected. This agrees with our principal assumption of the spectral embedding objective function in Section 2.5.1.4, that the graph is connected. Thus, in addition to addressing sparsity and heterogeneity issues,

Amini et al. (2013)'s regularization method ensures that the assumption that the graph is connected is satisfied. In this thesis, we use Amini et al. (2013)'s regularization method.

Let us define the *regularized degree normalized weighted adjacency matrix*

$$M = D_\tau^{-1/2} W_\tau D_\tau^{-1/2}, \tag{2.60}$$

where W_τ is the regularized weighted adjacency matrix with respect to the $n \times n$ weighted adjacency matrix, W, of G obtained by

$$W_\tau = W + \tau \mathbf{1}\mathbf{1}^T, \tag{2.61}$$

and $\mathbf{1}$ is an n-dimensional column vector containing all ones. Similar to the calculations in Equation A.2, we can show that L_τ and M have the same eigenvectors, and that their eigenvalues have the relationship $\lambda^M = 1 - \lambda^{L_\tau}$. Hence, adapting optimization functions 2.48 and 2.51, we can show that an optimal low dimensional embedding of vertices in a sparse graph is given by solving objective functions

$$\min_{U \in \mathbb{R}^{n \times k}} \text{trace}(U^T L_\tau U), \tag{2.62}$$

or

$$\max_{U \in \mathbb{R}^{n \times k}} \| U^T M U \|_F^2, \tag{2.63}$$

subject to $U^T U = I$.

In this section, we provided theoretical justification as to why the regularized degree normalized graph Laplacian, L_τ, or the regularized degree normalized weighted adjacency matrix, M, is the best representation matrix from which to obtain the embedded points for vertices in a graph. Given one of these representation matrices, our next step is to extract the principal eigenvectors. To achieve this, we employ *low-rank matrix approximation.*

2.5.1.6 Low-Rank Matrix Approximation for Dimensionality Reduction

From Sections 2.5.1.4 and 2.5.1.5, an embedding of a graph consists of the principal eigenvectors of its representation matrix. The low dimensional embedded points capture the underlying connectivity structure of the graph after filtering out noise and redundant information. In this section, we review a technique called *singular value decomposition* (SVD) that can be used to extract these eigenvectors from a given matrix. We further show how combining SVD with low-rank matrix approximation methods can be used to

automatically determine the *truncation dimension*, which is the number of eigenvectors to retain from the decomposition.

First of all, let us state some basics regarding the rank of a matrix. Consider a matrix $A \in \mathbb{R}^{n \times m}$ such that, $\text{rank}(A) = r$ and $n \geq m$. Then,

- Each of the m columns of A can be expressed as a linear combination of r different vectors, $\mathbf{c}_1, \ldots \mathbf{c}_r$, where $\mathbf{c}_i \in \mathbb{R}^n$ for $i = 1, 2, \ldots, r$.
- Each of the n rows of A can be expressed as a linear combination of r different vectors, $\mathbf{f}_1, \ldots \mathbf{f}_r$, where $\mathbf{f}_j \in \mathbb{R}^m$ for $j = 1, 2, \ldots, r$.
- Any matrix, $A \in \mathbb{R}^{n \times m}$, with $\text{rank}(A) = r$, can be expressed as a product of a matrix, $C \in \mathbb{R}^{n \times r}$, and a matrix $F \in \mathbb{R}^{m \times r}$ as

$$A = CF^T, \tag{2.64}$$

  where the columns of C are the linearly independent vectors, $\mathbf{c}_1, \ldots \mathbf{c}_r$, and the rows of F^T are the linearly independent vectors, $\mathbf{f}_1, \ldots \mathbf{f}_r$.

Decomposing A as a product of new matrices, as given in Equation 2.64, is called *rank factorization*. There are several different methods which can be used to perform rank factorization. The eigenvalue decomposition (EVD) (Equation 2.7) is one such method. In this thesis, we employ a method called *singular value decomposition* (SVD).

2.5.1.6.1 Singular Value Decomposition

SVD is a commonly used decomposition method due to its computational efficiency and ability to generalize the EVD to non-square matrices (Rohe et al., 2015).

Definition 2.2 (Singular Value Decomposition)**.** Let r be the rank of $n \times m$ matrix A. Then there exists a singular value decomposition of A of the form

$$A = U\Sigma V^T, \tag{2.65}$$

where

1. U is an $n \times r$ orthonormal matrix,
2. V is an $m \times r$ orthonormal matrix,
3. Σ is a $r \times r$ diagonal matrix, with non-negative diagonal entries $\Sigma_{i,i} = \sigma_i$ for $1 \leq i \leq r$.

The columns of U are the *left singular vectors* of A, the columns of V (equivalently, the rows of V^T) are the *right singular vectors* of A, and the entries in Σ are the *singular values* of A, which are usually ordered $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r$. Each singular vector (left and right) is associated with a singular value (Figure 2.12). For a matrix $A \in \mathbb{R}^{n \times m}$ with $\text{rank}(A) = r$, Eckart and Young (1936) showed that the best rank r approximation of A is obtained by its SVD given in Definition 2.2.

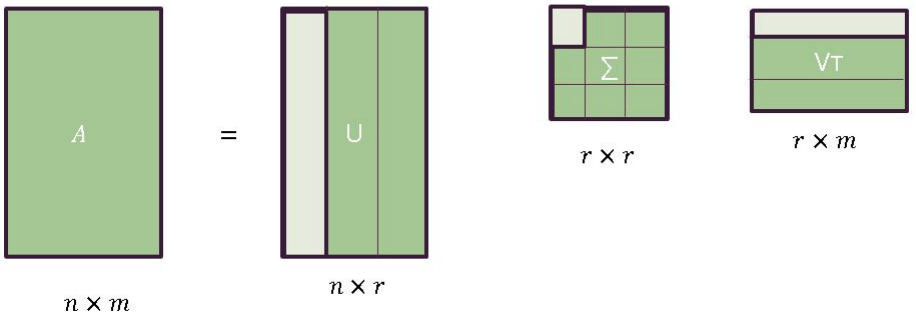


FIGURE 2.12: **Illustration of SVD**. Each singular value in Σ is associated with a left singular vector in U, and a right singular vector in V.

Below we state several properties of SVD, that are relevant to this thesis.

1. The right and left singular vectors of a matrix are unique upto an orthogonal transformation.

   Proof. Let $X = U$ and $Y = V\Sigma$. Then from Theorem 2.2,

   $$A = XY^T.$$

   Equally choose $\hat{X} = U\Gamma$ and $\hat{Y} = V\Sigma\Gamma$, where Γ is a $r \times r$ orthogonal matrix. Then,

   $$\begin{aligned} \hat{X}\hat{Y}^T &= U\Gamma\Gamma^T\Sigma V^T \\ &= A. \end{aligned} \tag{2.66}$$

   The orthogonally transformed left and right singular vectors reconstruct the original matrix. □

2. The matrices U and V contain the eigenvectors of the symmetric, positive semi-definite matrices AA^T and A^TA, respectively. Both AA^T and A^TA have the same eigenvalues, and these values are contained in the diagonal of Σ^2.

Proof. Let $A = U\Sigma V^T$ be the SVD of A. Then

$$\begin{aligned} AA^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\ &= U\Sigma \underbrace{V^T V}_{=I} \Sigma U^T \\ &= U\Sigma^2 U^T. \end{aligned} \tag{2.67}$$

Equation 2.67 is in the form of the EVD of AA^T (Equation 2.7). Similarly, it can be proved that $A^T A = V\Sigma^2 V^T$. □

3. Let $A \in \mathbb{R}^{n\times n}$ be a square, symmetric matrix. Then, the SVD of A is equivalent to its eigenvalue decomposition.

Proof. From Definition 2.2, $A = U\Sigma V^T$. For a symmetric matrix $A = A^T$. Then we have

$$\begin{aligned} A &= U\Sigma V^T \\ &= A^T \\ &= V\Sigma U^T \end{aligned}$$

which is satisfied if and only if $V = U$. Therefore,

$$A = U\Sigma U^T,$$

which is exactly the EVD of A (Equation 2.7). Thus, when the matrix is square and symmetric, the left and right singular vectors are the eigenvectors of A, and the singular values are its eigenvalues. □

2.5.1.6.2 Relationship between SVD and Matrix Low-rank Approximation

Let d denote the number of eigenvectors that we need to retain in the embedding. From our discussion in Section 2.5.1.4, we seek a low dimensional embedding, $U \in \mathbb{R}^{n\times d}$, of the representation matrix. This is exactly what is obtained as the left singular vectors if our representation matrix is of rank d. However, in reality, our representation matrix is of rank $r \gg d$. Thus, the usual practise is to truncate the SVD (Equation 2.65) at a low dimension d and obtain a low-rank approximation that best approximates the original representation matrix. Let $\hat{A} \in \mathbb{R}^{n\times m}$ be the low-rank approximation of matrix $A \in \mathbb{R}^{n\times m}$, obtained from the truncated SVD, where

$$\hat{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^T, \tag{2.68}$$

and $\tilde{U} \in \mathbb{R}^{n \times d}$, $\tilde{\Sigma} \in \mathbb{R}^{d \times d}$ and $\tilde{V} \in \mathbb{R}^{m \times d}$. By writing the SVD of A as a weighted sum of rank 1 components, we have

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T. \tag{2.69}$$

Since $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r$, when i increases, the contribution of the ith term becomes less important. Thus, by truncating the SVD at a lower rank-d, we have an approximation with error

$$\| A - \hat{A} \|_F^2 = \sum_{i=d+1}^{r} \sigma_i^2. \tag{2.70}$$

2.5.1.6.3 Deciding the Optimal Truncation Dimension - d

The truncation dimension, d, is usually chosen by plotting the singular values, $\sigma_1, \ldots, \sigma_r$, in order against their respective indices in a so-called *scree plot* (Cattell, 1966). The threshold, d, is determined by looking at the elbow or knee of the curve. Figure 2.13 shows an example of a scree plot. However, the scree plot method requires a human observer to determine d. Zhu and Ghodsi (2006) introduce an automatic method to find the elbow of the scree plot by maximizing a simple profile likelihood function. However, in real-world applications they show that it is often hard to observe a clear elbow or knee in the scree plot. Thus in practice, a method that relies heavily on the singular values may not always be useful.

In this thesis, we employ Achlioptas and McSherry (2007)'s low-rank matrix approximation method to get a good estimate of d. A brief explanation of this method is discussed in this section. Let us first introduce some basic concepts.

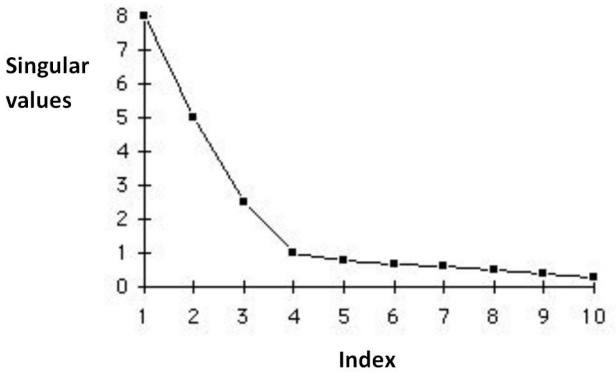


FIGURE 2.13: **Illustration of a scree plot**. The elbow or knee of the curve is at $d = 4$.

- A *linear trend* in a matrix, A, is the tendency of its rows to align with a unit vector, $\mathbf{v}$ (Achlioptas and McSherry, 2007). In order to measure this, A can be

multiplied by $\mathbf{v}$; each element of the n-dimensional vector, $A\mathbf{v}$, is the projection of the corresponding row of A onto $\mathbf{v}$. The strength of this linear trend is measured by $\| A\mathbf{v} \|_F$. With reference to Equation 2.12, the L_2 norm of A, $\| A \|_2$, gives the strongest linear trend in A. Computing a low-rank-d approximation, $\hat{A}$ (Equation 2.68), corresponds to extracting the d most significant linear trends from A.

- A matrix where some or all of its elements are random variables, is called a *random matrix* (Anderson et al., 2010). Initially, Wigner (1993) introduced the *semi-circle law*, describing the limiting eigenvalue distribution of a simple square, symmetric random matrix. His result was later extended to different types of random matrices. Achlioptas and McSherry (2007) propose one such extension and provide the following theorem on the spectral structure of non-symmetric random matrices.

  Theorem 2.3. *Let E be a random $n \times m$ matrix such that, each element, $E_{i,j} = r_{i,j}$, is an independent random variable with expected value zero, variance bounded by σ^2, and range, $[-K, K]$. For any $\alpha \geq 1, \epsilon > 0$, and $m + n \geq 20$, if*

  $$K \leq (\frac{4\epsilon}{4 + 3\epsilon})^3 \frac{\sigma\sqrt{m+n}}{log^3(m+n)}, \tag{2.71}$$

  then,

  $$\mathbb{P}\left[\| E \|_2 > (2 + \alpha + \epsilon)\sigma\sqrt{m+n}\right] < (m+n)^{-\alpha^2}.$$

  Let A be a real valued $n \times m$ matrix. Obtain $\tilde{A} \in \mathbb{R}^{n \times m}$ from A by altering the sign of each element in A with probability 1/2, that is, $\mathbb{P}[\tilde{A}_{i,j} = A_{i,j}] = \frac{1}{2}$ and $\mathbb{P}[\tilde{A}_{i,j} = -A_{i,j}] = \frac{1}{2}$, for all i, j. This process is called *random sign flipping*. Achlioptas and McSherry (2007) show that the random sign flipping of a matrix destroys any significant linear trends in A. Specifically, they show that the elements of $\tilde{A}$ are independent random variables with expected value zero, variance bounded by b^2, and range $[-b, b]$, where $b = \max A_{i,j}$. Using Theorem 2.3, the L2 norm, $\| \tilde{A} \|_2$, is bounded by $cb\sqrt{n+m}$, for some constant c. Thus, with high probability, the random sign flipping removes any significant linear trends in A.

From Equation 2.69, a rank r matrix, $A \in \mathbb{R}^{n \times m}$, can be expressed as $A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. Let R_d be the residual matrix after removing the low-rank-d approximation, $\hat{A}_d$, from A. We obtain R_d as

$$R_d = A - \hat{A}_d,$$

where $\hat{A}_d = \sum_{i=1}^{d} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. When d is the optimal truncation dimension, $\hat{A}_d$ contains all the important linear trends in A, and R_d will resemble a random matrix with no linear trends. In Figure 2.14, we illustrate this procedure using an adjacency matrix, A, from

a clear three block stochastic block model graph with 900 vertices. According to this figure, the optimal low-rank approximation, $\hat{A}_3$, contains all the important structure of A, while R_3 displays no significant structure. Hence, the optimal low-rank of A is three in this example. Achlioptas and McSherry (2007) test the L_2-norm of R_d to

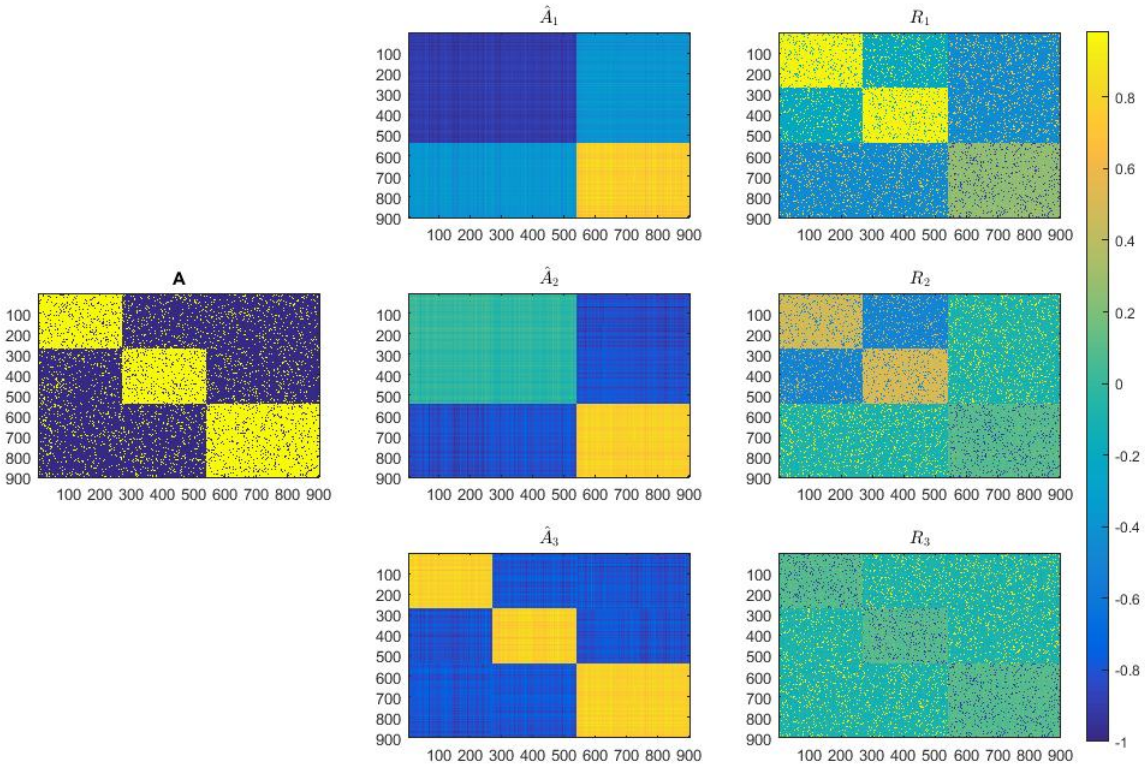


FIGURE 2.14: **Illustration of the low-rank matrix approximation process**. The adjacency matrix A has a clear three block structure. The first reconstruction $\hat{A}_1$ contains a portion of the structure from the original matrix A. The remaining structure is still visible in $R1$. Next, R_2 contains less structure than R_1, and $\hat{A}_2$ shows more structure than $\hat{A}_1$. Finally, R_3 contains hardly any structure, and resembles a random matrix, while $\hat{A}_3$ contains all the important structure of A.

determine the optimal truncation dimension, d. The truncation dimension, d, that produces a residual matrix, R_d, such that the L_2-norm of R_d is close to the L_2-norm of a random matrix is selected. Specifically, the dimension d such that $\| R_d \|_2 \approx \| \tilde{R}_d \|_2$, is selected as the optimal truncation dimension. Here, $\tilde{R}_d$ is the matrix obtained by random sign flipping of R_d. Achlioptas and McSherry (2007)'s low-rank matrix approximation method is a fast and computationally efficient method for matrices that have strong spectral structure, that is, matrices that depict a reasonable gap between the dth singular value, σ_d, and the $(d+1)$th singular value, σ_{d+1}.

There are other alternative approaches that can be used to determine the optimal truncation dimension, d. Jackson (1993) empirically evaluates the performance of several

dimension selection methods such as the Kaiser-Guttman method (Guttman, 1954), the bootstrapped Kaiser-Guttman method (Lambert et al., 1990), the broken-stick method, the test of sphericity (Cooley and Lohnes, 1971), and the proportion of total variance method (Jolliffe, 1972). Moreover, there exist some statistical approaches such as the linear regression approach by Zoski and Jurs (1996) that can be employed for the same purpose.

2.5.2 Spectral Embedding Methods for Graphs Represented by Tensors

Tensors provide an elegant way to represent the multiple edge attributes of a multi-graph (Nickel, 2013). An unweighted multi-graph consisting of n vertices and l different edge attributes can be represented in the form of an adjacency tensor, $\mathbf{A} \in \mathbb{R}^{n \times n \times l}$, with elements,

$$A_{i,j,k} = \begin{cases} 1, & \text{if } v_i \text{ is adjacenct to } v_j \text{ with respect to edge attribute } k, \\ 0, & \text{otherwise .} \end{cases} \tag{2.72}$$

In a weighted multi-graph, the edge between vertex i and vertex j with respect to edge attribute k will be assigned a non-negative weight, $W_{i,j,k}$. In this case, the weighted multi-graph can be encoded as an $n \times n \times l$ weighted adjacency tensor, $\mathbf{W}$, where $W_{i,j,k} = 0$ means vertices i and j are not connected in graph k.

In Section 2.5.1, we discussed the methodology behind obtaining a low dimensional representation of the vertices using matrix decomposition methods. In a similar manner, we can obtain a low dimensional representation of the vertices in a multi-graph using tensor decomposition methods. In Sections 2.5.2.2 and 2.5.2.3, we review the tensor decomposition approach used in this thesis. Prior to that, in Section 2.5.2.1, we provide a brief summary of tensor algebra and notations. A more detailed discussion on several tensor decomposition methods and applications can be found in Kolda and Bader (2009). The following discussions made in terms of $\mathbf{A}$ are also applicable to $\mathbf{W}$.

2.5.2.1 Relevant Tensor Algebra

A tensor is an N-way array. More formally, tensors are multilinear mappings over a set of N vector spaces, each of which possesses its own coordinate system. The *order* of a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, is N, which is also referred to as the number of modes. An element of $\mathbf{A}$ is denoted $A_{i_1,\ldots,i_n,\ldots,i_N}$ where $1 \leq i_n \leq I_n$. *Fibers* are one-dimensional sub-arrays of a tensor. The mode–n fibres of $\mathbf{A}$ are the I_n-dimensional

vectors obtained from $\mathbf{A}$ by varying index i_n, while keeping other indices fixed. The fibers of a three-dimensional tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, will be denoted by $A_{.,j,k}$ (mode–1 fiber), $A_{i,.,k}$ (mode–2 fiber), and $A_{i,j,.}$ (mode–3 fiber), where $i \in \{1, \ldots, I_1\}$, $j \in \{1, \ldots, I_2\}$, and $k \in \{1, \ldots, I_3\}$ (Figure 2.15).

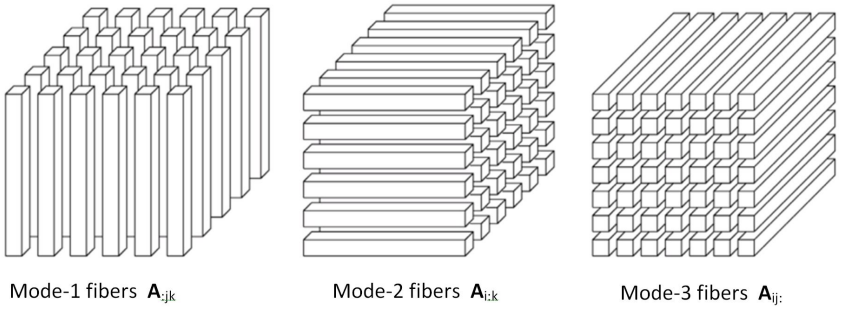


FIGURE 2.15: **Fibers of a three-mode tensor**. Mode-1, mode–2 and mode–3 fibers are also called as columns, rows and tubes respectively.

Slices are two-dimensional sub-arrays of a tensor defined by fixing all but two indices. The slices of a third order tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, are denoted by $A_{i,.,.}$ (mode–1 slice), $A_{.,j,.}$ (mode–2 slice), and $A_{.,.,k}$ (mode–3 slice) where $i \in \{1, \ldots, I_1\}$, $j \in \{1, \ldots, I_2\}$, and $k \in \{1, \ldots, I_3\}$ (Figure 2.16). The mode–3 slice, $A_{.,.,k}$, which encodes the connections in the graph corresponding to the kth edge attribute, plays a special role in Chapter 4 of this thesis.

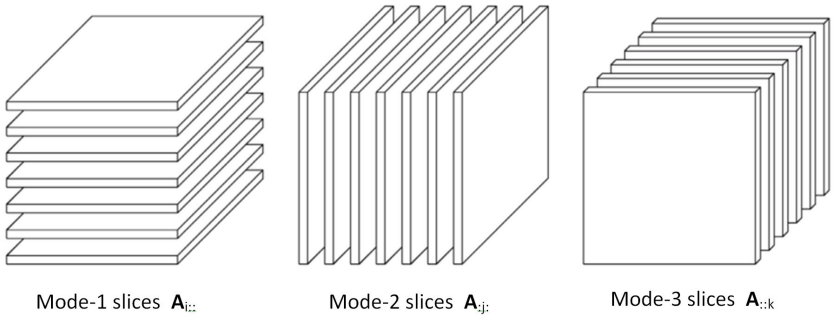


FIGURE 2.16: **Slices of a three-mode tensor**. Mode-1, mode–2 and mode–3 slices are also called as horizontal, lateral and frontal slices respectively.

2.5.2.1.1 The Frobenius Norm of a Tensor

The Frobenius norm of a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, is the square root of the sum of the squares of all its elements, that is,

$$\| \mathbf{A} \|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} A^2_{i_1,\ldots,i_n,\ldots,i_N}}. \tag{2.73}$$

2.5.2.1.2 Matricization: Transforming a Tensor into a Matrix

Matricization is the process of unfolding or flattening of a tensor by reordering its elements to form a matrix. The mode–n matricization of a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, is denoted by $\mathbf{A}_{(n)}$ and has dimensions $I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_N)$. The columns of $\mathbf{A}_{(n)}$ are the mode–n *fibres* of $\mathbf{A}$. In Figure 2.17, we show the mode–1, mode–2, and mode–3 matricizations of a three dimensional tensor.

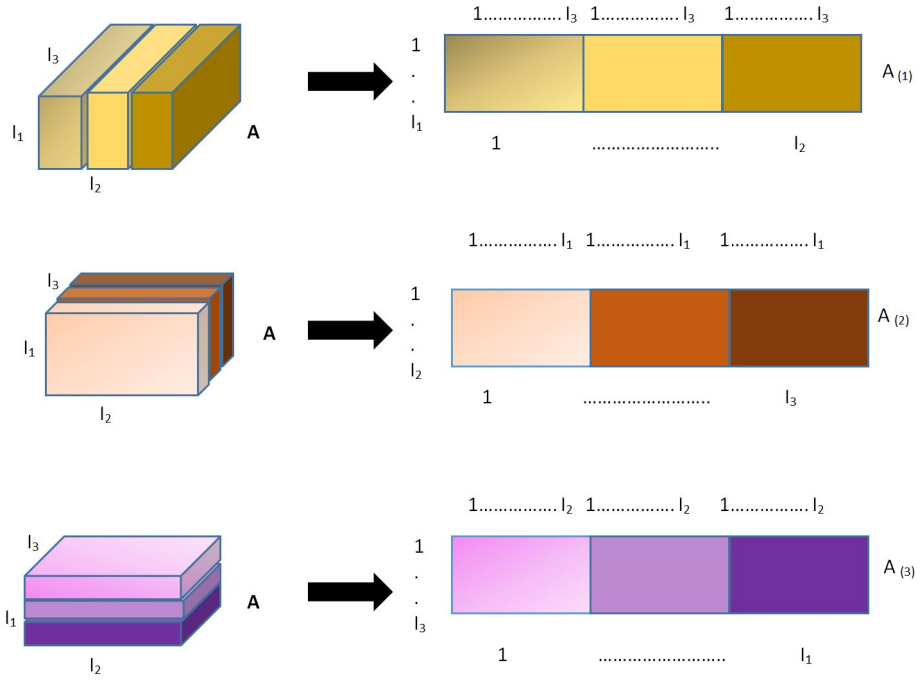


FIGURE 2.17: **Illustration of flattening of a three-mode tensor**. The tensor can be flattened in three ways to obtain matrices, where the columns contain the mode–n fibres.

2.5.2.1.3 Tensor Multiplication: The Mode-n Product

The *mode–n product* of a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, by a matrix, $M \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathbf{A} \times_n M$, is also a tensor of size, $I_1 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots I_N$. Element-wise, we have,

$$(\mathbf{A} \times_n M)_{I_1,\ldots,I_{n-1},J_n,I_{n+1},\ldots I_N} = \sum_{i_n=1}^{I_n} \mathbf{A}_{i_1,\ldots i_{n-1},i_n,i_{n+1},\ldots,i_N} M_{j_n,i_n}.$$

In terms of flattened matrices, the mode–n product is given by $\mathbf{B}_{(n)} = M\mathbf{A}_{(n)}$. For distinct modes in a series of multiplications, the order of the multiplication is irrelevant,

that is,

$$\mathbf{A} \times_m M \times_n P = \mathbf{A} \times_n M \times_m P \quad (m \neq n).$$

If the modes are the same, then

$$\mathbf{A} \times_n M \times_n P = \mathbf{A} \times_n (MP).$$

A matrix $A \in \mathbb{R}^{I_1 \times I_2}$ is a two dimensional array that possess two vector spaces, a row space and a column space. In Section 2.5.1, we discussed how these two spaces are orthogonalized using SVD, by decomposing it as $A = USV^T$, where $U \in \mathbb{R}^{I_1 \times J_1}$ is the orthogonalized column space, $V \in \mathbb{R}^{I_2 \times J_2}$, is the orthogonalized row space, and $S \in \mathbb{R}^{J_1 \times J_2}$, is a diagonal singular value matrix. A tensor, on the other hand, is an n-dimensional array that possesses $n > 2$ vector spaces. In Sections 2.5.2.2 and 2.5.2.3, we extend the ideas behind matrix decompositions to tensors. Prior to that, we define several terms that are relevant to our discussion.

2.5.2.1.4 The n-Rank

Let $\mathbf{A}$ be the tensor of size $I_1 \times I_2 \times \ldots I_N$. Then, the *n-rank* of $\mathbf{A}$ denoted by $\text{rank}_n(\mathbf{A})$, is the column rank of $\mathbf{A}_{(n)}$. In other words, the n-rank is the dimension of the subspace spanned by the mode–n fibers (Figure 2.15).

2.5.2.1.5 Multilinear Rank

If we denote $R_n = \text{rank}_n(\mathbf{A})$, for $n \in \{1, 2, \ldots N\}$, then we can say that $\mathbf{A}$ is a $\text{rank} - (R_1, R_2, \ldots, R_N)$ tensor. This is called the *multilinear rank* of tensor $\mathbf{A}$. It should be noted that the values of $R_1, R_2, \ldots R_N$ can be different for $N \geq 3$. Hence, the multilinear rank is different from the matrix rank (Section 2.2.1).

2.5.2.2 Tucker Decomposition

Tucker decomposition was developed by Ledyard Tucker (Tucker, 1966) by generalizing SVD inorder to decompose tensors. It decomposes a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$, in to a core tensor, $\mathbf{S} \in \mathbb{R}^{R_1 \times R_2 \times \ldots R_N}$, multiplied by the column-wise orthonormal factor matrices, $U^{(n)} = [\mathbf{u}_1^{(n)}, \mathbf{u}_2^{(n)}, \ldots \mathbf{u}_{R_n}^{(n)}] \in \mathbb{R}^{I_n \times R_n}$, along each mode as

$$\mathbf{A} = \mathbf{S} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_N U^{(N)}. \tag{2.74}$$

From Kolda (2006), Equation 2.74 can be rewritten in tensor notation as

$$\mathbf{A} = [\![\mathbf{S}; U^{(1)}, U^{(2)}, \ldots, U^{(N)}]\!]. \tag{2.75}$$

We illustrate the Tucker decomposition for a third order tensor in Figure 2.18.

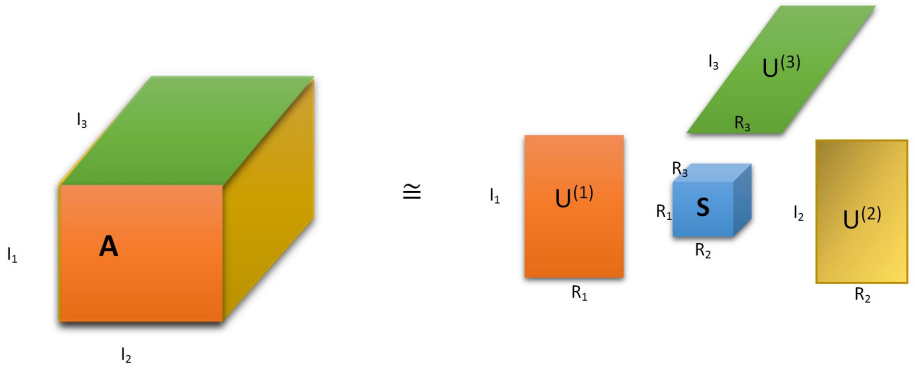


FIGURE 2.18: **Tucker decomposition of a three mode tensor**. The columns spaces of $U^{(1)}$, $U^{(2)}$, $U^{(3)}$ represent orthogonal subspaces for the three modes in the tensor. The core tensor, $\mathbf{S}$, is non-diagonal and accounts for all possible interactions among the tensor components.

2.5.2.2.1 Uniqueness

Tucker decompositions are generally not unique, that is, the factor matrices, $U^{(n)}$, can be modified without affecting the overall fit. Let $\Gamma^{(n)} \in \mathbb{R}^{R_n \times R_n}$ be the non-singular matrix that is applied to modify the mode-n factor matrix. Then from Equation 2.75 we have,

$$\begin{aligned} \mathbf{A} &= [\![\mathbf{S}; U^{(1)}, U^{(2)}, \ldots, U^{(N)}]\!], \\ &= [\![\mathbf{S} \times_1 {\Gamma^{(1)}}^{-1} \times_2 {\Gamma^{(2)}}^{-1} \ldots \times_N {\Gamma^{(N)}}^{-1}; U^{(1)}\Gamma^{(1)}, U^{(2)}\Gamma^{(2)}, \ldots, U^{(N)}\Gamma^{(N)}]\!]. \end{aligned} \tag{2.76}$$

In other words, any factor matrix can be post multiplied by any non-singular matrix without affecting the overall fit, as long as the modifications are compensated for by the core tensor.

2.5.2.3 Computing Tucker Decomposition using Higher Order Singular Value Decomposition (HOSVD)

Tucker (1966) introduced the higher order singular value decomposition (HOSVD) method to compute the Tucker decomposition of a tensor. The method finds the mode–n factor matrices as those columns that best captures the variation of mode–n, independent

of other modes. De Lathauwer et al. (2000a) provide theoretical evidence that the orthonormal columns of each factor matrix can be obtained from the singular value decomposition of the corresponding mode–n matricized tensor, $\mathbf{A}_{(n)}$. In this section we outline De Lathauwer et al. (2000a)'s method.

Let $S_{i_n=\alpha}$ be the corresponding slice of the core tensor $\mathbf{S}$ (equation 2.74), obtained by fixing the nth mode to α. The HOSVD method assumes the following properties of $\mathbf{S}$:

1. *All-orthogonality*: Two slices, $S_{i_n=\alpha}$ and $S_{i_n=\beta}$, are orthogonal for all possible values of n, α and β subject to $\alpha \neq \beta$:

$$\langle S_{i_n=\alpha}, S_{i_n=\beta} \rangle = 0 \quad \text{where} \quad \alpha \neq \beta. \tag{2.77}$$

2. *Ordering*: Let the Frobenius norms, $\| S_{i_n=i} \|_F$, symbolized by $\sigma_i^{(n)}$ be the n-mode singular values of $\mathbf{A}$. Then,

$$\| S_{i_n=1} \|_F \geq \| S_{i_n=2} \|_F \geq \ldots \| S_{i_n=R_n} \|_F \geq 0, \tag{2.78}$$

   for all possible values of n. Equivalently this can be written as

$$\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \ldots \geq \sigma_{R_n}^{(n)} \geq 0.$$

Via the Kronecker products (Equation 2.13), the Tucker decomposition shown in Equation 2.75 can be expressed in a matrix form as

$$\mathbf{A}_{(n)} = U^{(n)} \mathbf{S}_{(n)} \left(U^{(n+1)} \otimes U^{(n+2)} \otimes \ldots U^{(N)} \otimes U^{(1)} \otimes U^{(2)} \otimes \ldots \otimes U^{(n-1)} \right)^T. \tag{2.79}$$

De Lathauwer et al. (2000a) show that this mode–n matricized tensor can be represented as a singular value decomposition given by

$$\mathbf{A}_{(n)} = U^{(n)} \Sigma^{(n)} V^{(n)^T}, \tag{2.80}$$

where $\Sigma^{(n)} \in \mathbb{R}^{R_n \times R_n}$ is a diagonal matrix with diagonal values, $\sigma_1^{(n)}, \sigma_2^{(n)}, \ldots, \sigma_{R_n}^{(n)}$, and

$$V^{(n)^T} = \bar{\mathbf{S}}_{(n)} \left(U^{(n+1)} \otimes U^{(n+2)} \otimes \ldots U^{(N)} \otimes U^{(1)} \otimes U^{(2)} \otimes \ldots \otimes U^{(n-1)} \right).$$

Here, $\bar{\mathbf{S}}_{(n)}$ is a normalized version of $\mathbf{S}_{(n)}$, with rows scaled to unit length as

$$\mathbf{S}_{(n)} = \Sigma^{(n)} \bar{\mathbf{S}}_{(n)}.$$

So, the factor matrices, $U^{(n)} \in \mathbb{R}^{I_n \times R_n}$ (for $n \in \{1, 2, \ldots, N\}$), can be obtained by performing SVD on each $\mathbf{A}_{(n)}$, and extracting the matrix of left singular vectors. In Figure 2.19, we illustrate this for a three mode tensor. Once factor matrices are obtained for all modes, the core tensor, $\mathbf{S}$, can be calculated as

$$\mathbf{S} = \mathbf{A} \times_1 U^{(1)^T} \times_2 U^{(2)^T} \ldots \times_N U^{(N)^T}. \tag{2.81}$$

This procedure of calculating the factor matrices using Equation 2.80, and then calculating $\mathbf{S}$ using Equation 2.81, is the HOSVD or multilinear SVD of a tensor.

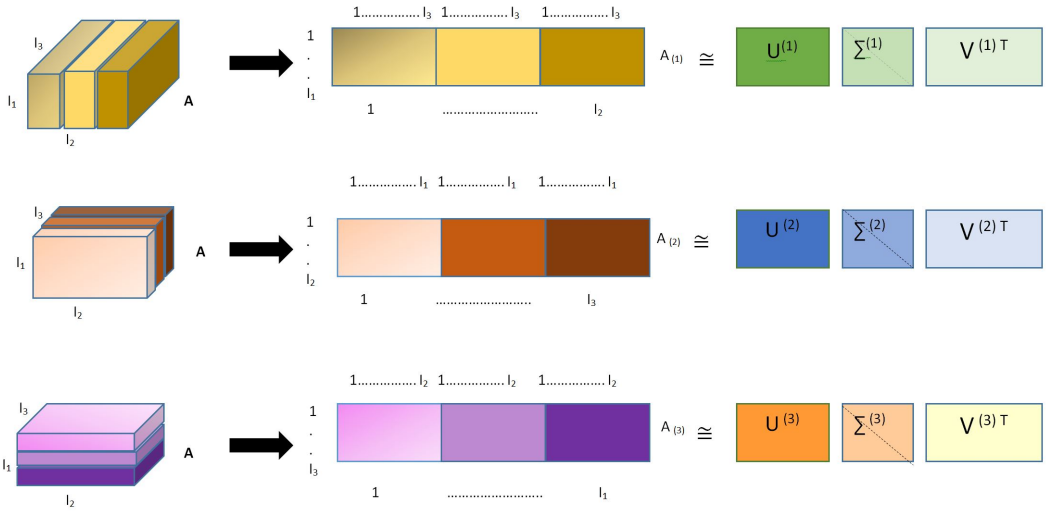


FIGURE 2.19: **Illustration of HOSVD of a three-mode tensor**. The tensor can be flattened in three ways to obtain matrices, where the columns contain the mode–n fibres. SVD is then applied on each flattened matrix. The left singular vectors from each decomposition define the factor matrices $U^{(1)}, U^{(2)}$, and $U^{(3)}$.

2.5.2.3.1 Properties of HOSVD

1. **n-rank**

   Let R_n be equal to the highest index for which $\| S_{i_n=R_n} \|_F > 0$, then one has

   $$R_n = \text{rank}_n(\mathbf{A}) = \text{rank}(\mathbf{A}_{(n)}). \tag{2.82}$$

   This holds for $n \in \{1, 2, \ldots, N\}$. The diagonal of each $\Sigma^{(n)}$ contains the mode–n singular values. The number of non-zero mode–n singular values equals the n-rank of the tensor.

2. **uniqueness**

   When the mode–n singular values are different, the mode–n singular vectors are uniquely determined up to an orthogonal transformation (see Section 2.5.2.2.1).

3. **norm**

   Let the HOSVD of $\mathbf{A}$ be given as in Equations 2.80 and 2.81. Then,

$$\| \mathbf{A} \|_F^2 = \sum_{i=1}^{R_1} (\sigma_i^{(1)})^2 = \ldots = \sum_{i=1}^{R_N} (\sigma_i^{(N)})^2 = \| \mathbf{S} \|_F^2 . \tag{2.83}$$

Further details and proof can be found in De Lathauwer et al. (2000a).

2.5.2.3.2 Low-Rank Approximation of a Tensor

For a tensor, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$, the low-rank approximation, $\hat{\mathbf{A}}$, is obtained by minimizing the following least squares cost function,

$$E = \| \mathbf{A} - \hat{\mathbf{A}} \|_F^2, \tag{2.84}$$

where

$$\hat{\mathbf{A}} = \mathbf{S} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_N U^{(N)}. \tag{2.85}$$

Here, each factor matrix, $U^{(n)} \in \mathbb{R}^{I_n \times d_n}$, with $d_n < I_n$, is the truncated factor matrices with orthonormal columns, and $\mathbf{S} \in \mathbb{R}^{d_1 \times d_2 \times \ldots d_N}$ is the core tensor. In Section 2.5.1.6, we discussed how we discard the singular vectors corresponding to the less significant singular values to obtain a low-rank approximation of a matrix. Similarly, in HOSVD, we obtain a low-rank approximation of a tensor by discarding the smallest mode–n singular values. We also discard the mode–n singular vectors and the slices of the core tensor associated with the smallest mode–n singular values (Figure 2.20).

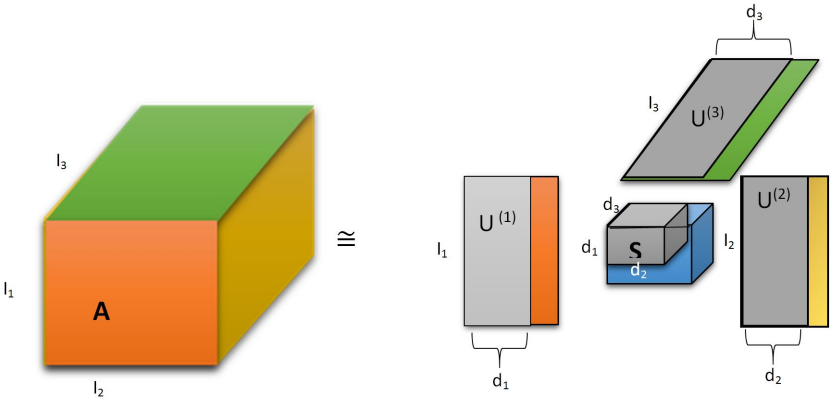


FIGURE 2.20: **Illustration of low-rank approximation of a mode–3 tensor**. The low-rank approximation is obtained by discarding the singular vectors, and the slices of the core tensor corresponding to the smallest mode–n singular values.

Let the HOSVD of $\mathbf{A}$ be calculated as discussed in Equation 2.80, and let the mode–n rank of $\mathbf{A}$ be equal to R_n. Define a tensor $\hat{\mathbf{A}}$, by discarding the smallest mode–n singular values, $\sigma^{(n)}_{d_{n+1}}, \sigma^{(n)}_{d_{n+2}}, \ldots, \sigma^{(n)}_{R_n}$, for given values of d_n. The approximation error is given by

$$\| \mathbf{A} - \hat{\mathbf{A}} \|_F^2 \leq \sum_{i_1=d_1+1}^{R_1} \sigma_{i_1}^{(1)^2} + \sum_{i_2=d_2+1}^{R_2} \sigma_{i_2}^{(2)^2} + \ldots \sum_{i_N=d_N+1}^{R_N} \sigma_{i_N}^{(N)^2}. \tag{2.86}$$

It is important to note that although the sum of the discarded singular values gives the approximation error for SVD (Equation 2.70), the sum of the discarded singular values only provides an upper bound on the approximation error for HOSVD. This bound is given in Harshman (1970).

However, the tensor, $\hat{\mathbf{A}}$, obtained using HOSVD is in general not the best possible approximation for $\mathbf{A}$. Alternatively, in order to find the best low-rank approximation of a tensor, one approach is to use the *Higher Order Orthogonal Iteration* (HOOI) method (De Lathauwer et al., 2000b; ten Berge et al., 1987). Nevertheless, the HOSVD approximation is easy to implement and is still considered as a good approximation (De Lathauwer et al., 2000a). Thus, in this thesis, we obtain $\hat{\mathbf{A}}$ using the HOSVD method. In Chapter 4, we use the mode-1 truncated factor matrix, $U^{(1)} \in \mathbb{R}^{I_1 \times d_1}$, as the embedding of the multi-graph that summarizes the behaviour of entities in a multi-view network.

In Sections 2.5.1 and 2.5.2, we covered the methodology behind obtaining a spectral embedding of the vertices in a graph represented by a matrix, and a multi-graph represented by a tensor, respectively. The goal of this thesis is change detection. We detect changes involving vertices by evaluating the dissimilarity between embeddings extracted from graphs at different time instants. In the next section, we review the method we use for this purpose.

2.6 Procrustes Analysis

Procrustes analysis is a statistical technique that is used to compare two or more objects represented as matrices. More precisely, Procrustes analysis involves a least squares optimization technique that directly estimates and performs Euclidean similarity transformations on a set of matrices up to their maximal agreement. The Euclidean similarity transformations, (i) *translation* (Equation 2.87), (ii) *isomorphic scaling* (Equation 2.88), (iii) *rotation* (Equation 2.89), and (iv) *reflection* (Equation 2.90), preserve the *shape* of the object. The shape is all the geometrical information that remains after Euclidean similarity transformations are removed from an object. Procrustes methods compare the shapes of objects in order to measure dissimilarities between them.

In order to obtain the shape of an object, it is convenient to remove Euclidean similarity transformations one at a time. Let us consider a sketch of a hand as an object. A finite set of point coordinates on the surface of the hand (Figure 2.21 (A)) are called *landmarks*. The set of n landmarks in d dimensions on a particular object can be represented as a $n \times d$ matrix, X. Below, we define the four types of Euclidean similarity transformations that are used in Procrustes analysis.

1. *Translation*

   The translation step moves an object to a preferred coordinate by pre-multiplying by a suitable matrix. Usually, the origin, $(0, 0)$, of the coordinate system is preferred. The resulting *centred matrix*, $X_c \in \mathbb{R}^{n \times d}$, is given by

   $$X_c = CX, \tag{2.87}$$

   where $C = I - \frac{1}{n}\mathbf{j}\mathbf{j}^T$, I is an $n \times n$ identity matrix, and $\mathbf{j}$ is an $n-$dimensional vector of ones.

2. *Isomorphic scaling*

   Scaling is the manipulation technique that makes the object smaller or larger. Before calculating a distance between two shapes, it is necessary to standardize for size. Thus, in this thesis, we scale by dividing by the size or Frobenius norm. The resulting *pre-shape*, $\widetilde{X} \in \mathbb{R}^{n \times d}$, is given by

   $$\widetilde{X} = \frac{X_c}{\| X_c \|_F}. \tag{2.88}$$

3. *Rotation*

   The rotation of a matrix is given by the post-multiplication of the a matrix by an orthogonal matrix, $\tilde{\Gamma} \in \mathbb{R}^{d \times d}$, such that $\tilde{\Gamma}^T\tilde{\Gamma} = \tilde{\Gamma}\tilde{\Gamma}^T = I_d$, and the determinant, $|\tilde{\Gamma}| = +1$. When rotation is applied to the pre-shape, we obtain the shape of the object, $\widehat{X} \in \mathbb{R}^{n \times d}$, given by

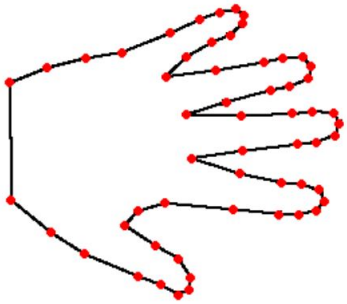
   $$\widehat{X} = \widetilde{X}\tilde{\Gamma}. \tag{2.89}$$

4. *Reflection*

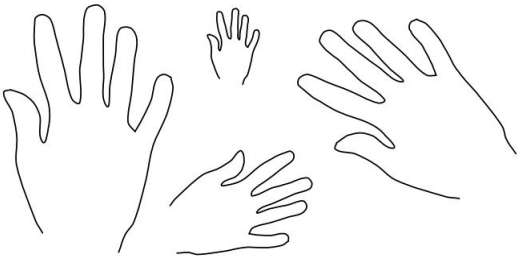
   In some situations it is suitable to remove both rotation and reflection transformations from the object. The reflection of a matrix is given by the post-multiplication of the original matrix by an orthogonal matrix, $\Gamma \in \mathbb{R}^{d \times d}$, such that $\Gamma^T\Gamma = \Gamma\Gamma^T = I_d$, and the determinant, $|\Gamma| = \pm 1$. When reflection is applied to the pre-shape, we obtain the reflection shape, $\widehat{X} \in \mathbb{R}^{n \times d}$, given by

   $$\widehat{X} = \widetilde{X}\Gamma. \tag{2.90}$$

In Figure 2.21 (B), we illustrate the resulting copies of the hand in Figure 2.21 (A), after filtering out different Euclidean similarity transformations.



(A) Illustration of an object (hand) being represented as a set of landmarks.



(B) Four copies of the same object under different Euclidean similarity transformations.

FIGURE 2.21: **Illustration of an object and similarity transformations (taken from Stegmann and Gomez (2002) with permission).** An object is represented by a set of landmarks. By applying Euclidean similarity transformations, we can represent the original object in different ways.

Procrustes analysis has many variations and forms. In this thesis, we use Procrustes analysis to compare embeddings across time instants by adjusting for scale, rotation and reflection transformations. Thus, we employ orthogonal Procrustes analysis methods. In Sections 2.6.1 and 2.6.2, we provide a brief review of this method based on Dryden and Mardia (1998), Ross (2004) and Borg and Groenen (2005).

2.6.1 Orthogonal Procrustes Analysis

Schönemann (1966) presents the least squares solution to the to the orthogonal Procrustes problem of fitting a pre-shape, $\widetilde{Y} \in \mathbb{R}^{n \times d}$, to another pre-shape, $\widetilde{X} \in \mathbb{R}^{n \times d}$, under the choice of an unknown orthogonal transformation, Γ, in such a way as to minimize the sum of squares of the residual matrix,

$$\| \widetilde{X} - \widetilde{Y}\Gamma \|_F^2, \tag{2.91}$$

subject to $\Gamma^T\Gamma = I$.

Since $\| X \|_F^2 = \text{trace}\{X^T X\}$, we have

$$\begin{aligned} \| \widetilde{X} - \widetilde{Y}\Gamma \|_F^2 &= \text{trace}\{(\widetilde{X} - \widetilde{Y}\Gamma)^T(\widetilde{X} - \widetilde{Y}\Gamma)\} \\ &= \text{trace}\{\widetilde{X}^T\widetilde{X}\} + \text{trace}\{\widetilde{Y}^T\widetilde{Y}\} - 2\text{trace}\{\widetilde{X}^T\widetilde{Y}\Gamma\}. \end{aligned} \tag{2.92}$$

The first two terms on the right hand side of Equation 2.92 do not contain Γ. Thus, a least squares solution to Equation 2.91 should maximize the term, $\text{trace}\{\widetilde{X}^T\widetilde{Y}\Gamma\}$.

By using the singular value decomposition, $\widetilde{X}^T\widetilde{Y} = U\Sigma V^T$, and the cyclic property of matrix trace we have,

$$\begin{aligned} \text{trace}\{\widetilde{X}^T\widetilde{Y}\Gamma\} &= \text{trace}\{U\Sigma V^T\Gamma\} \\ &= \text{trace}\{\Sigma V^T\Gamma U\} \\ &= \text{trace}\{\Sigma H\}, \end{aligned}$$

where $H = V^T\Gamma U$. Here, H is an orthogonal matrix[15] with dimensions $r \times r$, where r is the rank of $\widetilde{X}^T\widetilde{Y}$. Thus, we have

$$\text{trace}\{\Sigma H\} = \sum_{i=1}^{r} \Sigma_{i,i} H_{i,i}.$$

Recall from Section 2.5.1.6.1 that $\Sigma_{i,i}$ are non-negative numbers. Thus, trace$\{\Sigma H\}$ is maximized when $H_{i,i} = 1$ for $i \in \{1, \ldots, r\}$. Thus, we have

$$\begin{aligned} H &= I \quad \text{and} \\ V^T\Gamma U &= I. \end{aligned} \tag{2.93}$$

From Equation 2.93, the least squares estimate $\hat{\Gamma}$ that solves problem 2.91 is,

$$\hat{\Gamma} = VU^T. \tag{2.94}$$

After estimating $\hat{\Gamma}$ we calculate the *orthogonal Procrustes fit* of Y onto X.

Definition 2.4 (Full Procrustes Fit)**.** The orthogonal Procrustes fit of Y onto X is defined as

$$\hat{Y} = \widetilde{Y}\hat{\Gamma}. \tag{2.95}$$

Note that we initially obtain the pre-shapes, $\widetilde{X}$ and $\widetilde{Y}$, before matching the objects under rotation or reflection in order to make sure that X and Y are invariant under translation and scale. After doing the Procrustes superimposition of Y onto X as in Equation 2.91, the Procrustes distance between X and Y can be calculated as

$$d_{OPA} = \| \widetilde{X} - \widetilde{Y}\hat{\Gamma} \|_F . \tag{2.96}$$

The above procedure is limited in its application as it can only match two matrices. Generalized orthogonal Procrustes analysis can be applied in the presence of $k \geq 2$ matrices.

[15] H the product of orthogonal matrices.

2.6.2 Generalized Orthogonal Procrustes Analysis (GPA)

The generalized orthogonal Procrustes analysis is used to superimpose a set of matrices under rotation and reflection in order to minimize the sum of pairwise Euclidean distances. Let $X_1, \ldots, X_m$, be the m matrices that need to be superimposed, where each has dimension $n \times d$. GPA minimizes the least squares objective function

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} \parallel \widetilde{X}_i \Gamma_i - \widetilde{X}_j \Gamma_j \parallel_F^2, \tag{2.97}$$

where,
$\widetilde{X}_i$ is the pre-shape corresponding to X_i obtained from Equation 2.88, and
Γ_i is the $d \times d$ orthogonal rotation/reflection matrix corresponding to X_i, such that, $\Gamma_i^T \Gamma_i = I$.

An iterative algorithm is required to minimize the GPA objective function 2.97, because there is no direct analytical solution. Borg and Groenen (2005) describe different approaches that can be taken for this purpose. In this thesis, we employ the approach that uses the concept of a centroid, $\mu = \frac{1}{m} \sum_{i=1}^{m} \tilde{\tilde{X}}_i$, where $\tilde{\tilde{X}}_i = \widetilde{X}_i \Gamma_i$. The GPA objective function 2.97 is equivalent to minimizing

$$m \sum_{i=1}^{m} \parallel \tilde{\tilde{X}}_i - \mu \parallel_F^2, \tag{2.98}$$

by updating $\tilde{\tilde{X}}_i$ and μ one at a time while keeping the other fixed. This form of GPA aligns each matrix to the centroid matrix μ. Therefore, μ can be thought of as the matrix that summarises a common representation for all the optimally transformed $\tilde{\tilde{X}}_i$s. Below we show how objective function 2.97 is equivalent to 2.98:

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=i+1}^{m} \parallel \tilde{\tilde{X}}_i - \tilde{\tilde{X}}_j \parallel_F^2 &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \text{trace}\{[\tilde{\tilde{X}}_i - \tilde{\tilde{X}}_j]^T [\tilde{\tilde{X}}_i - \tilde{\tilde{X}}_j]\} \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \text{trace}\{[\tilde{\tilde{X}}_i - \tilde{\tilde{X}}_j]^T [\tilde{\tilde{X}}_i - \tilde{\tilde{X}}_j]\} \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \text{trace}\{\tilde{\tilde{X}}_i^T \tilde{\tilde{X}}_i\} + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \text{trace}\{\tilde{\tilde{X}}_j^T \tilde{\tilde{X}}_j\} \\
&\quad - \sum_{i=1}^{m} \sum_{j=1}^{m} \text{trace}\{\tilde{\tilde{X}}_i^T \tilde{\tilde{X}}_j\}.
\end{aligned} \tag{2.99}$$

By summing the first two terms in the right hand side of Equation 2.99 we have

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\text{trace}\{\tilde{\tilde{X}}_j^T\tilde{\tilde{X}}_j\}=m\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}. \tag{2.100}$$

As $\tilde{\tilde{X}}_i$ in the last term of Equation 2.100 does not depend on j we have

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_j\} &= \sum_{i=1}^{m}\text{trace}\left\{\sum_{j=1}^{m}\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_j\right\}\\ &= m\sum_{i=1}^{m}\text{trace}\left\{\tilde{\tilde{X}}_i^T\left(\frac{1}{m}\sum_{j=1}^{m}\tilde{\tilde{X}}_j\right)\right\}\\ &= m\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}. \end{aligned}\tag{2.101}$$

By substituting the results from Equations 2.100 and 2.101 in Equation 2.99 we have

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=i+1}^{m}\|\tilde{\tilde{X}}_i-\tilde{\tilde{X}}_j\|_F^2 &= m\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}-m\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}-\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}-2\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\sum_{i=1}^{m}\text{trace}\{\mu^T\tilde{\tilde{X}}_i\}-2\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\text{trace}\{\sum_{i=1}^{m}\mu^T\tilde{\tilde{X}}_i\}-2\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\text{trace}\{\mu^T m\mu\}-2\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\left(\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}+\sum_{i=1}^{m}\text{trace}\{\mu^T\mu\}-2\sum_{i=1}^{m}\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}\right)\\ &= m\sum_{i=1}^{m}\left(\text{trace}\{\tilde{\tilde{X}}_i^T\tilde{\tilde{X}}_i\}-2\text{trace}\{\tilde{\tilde{X}}_i^T\mu\}+\text{trace}\{\mu^T\mu\}\right)\\ &= m\sum_{i=1}^{m}\text{trace}\{[\tilde{\tilde{X}}_i-\mu]^T[\tilde{\tilde{X}}_i-\mu]\}. \end{aligned}\tag{2.102}$$

Thus, objective functions 2.97 and 2.98 are equivalent. Minimization of objective function 2.98 is preferable due to its simplicity compared to 2.97. The final solution for the centroid μ shows the final coordinates of the mean matrix that has maximal agreement with respect to the least squares objective function 2.98. Each unknown Euclidean

similarity transformation parameter, $\hat{\Gamma}_i$, can be determined by means of orthogonal Procrustes of each $\widetilde{X}_i$ with μ. In this thesis, we employ the iterative approach in Cootes et al. (2004) and Stegmann and Gomez (2002) to do GPA.

2.6.3 Relevant Applications of Procrustes Analysis

To the best of our knowledge, Procrustes methods have never been used for change detection in dynamic networks. However, Procrustes methods are commonly used for shape recognition and object matching in applications related to video processing. Vaswani et al. (2005) model the behaviour of a group of moving objects by treating them as a time sequence of matrices denoting landmarks at each time instant. They use Procrustes methods to detect irregular behaviour of the object. Rougier et al. (2008) represent a moving human being as a time sequence of matrices, and use Procrustes methods to detect unusual events such as falling.

Although not precisely related to change detection, Tang et al. (2014) introduce a hypothesis testing procedure based on orthogonal Procrustes matching to compare two embeddings obtained from two random dot product graphs (RDPGs) (Section 2.3.2.2). Specifically, based on the adjacency spectral graph embeddings obtained from two finite dimensional RDPGs on the same set of vertices, a test is developed to investigate whether the graphs have the same generating latent positions or whether the generating latent positions are scaled or diagonal transformations of one another. Tang et al. (2014)'s test can be applied to compare two embeddings obtained from the binary adjacency matrix of two RDPGs. Inspired by their idea, we utilize Procrustes techniques for change detection across multiple graphs.

Chapter 3

Change Detection in Dynamic Networks using Procrustes Analysis

As described in Chapter 1, *change detection* is the process of continuously monitoring a dynamic network for deviations in entities and their relationship structure. Given a dynamic network conceptualized as a time sequence of undirected, weighted graphs, we address the problem of detecting vertex-based changes at each time instant. Detecting vertex-based changes is important in areas such as fraud detection, cyber intrusion detection and spam detection. For example, consider the time varying email communications between a set of employees in an organisation. A sudden collaboration between a set of employees who rarely communicated during the recent past, may indicate some unusual motivation or a major event involving the organisation (Sricharan and Das, 2014). Such changes in entity behaviour can be detected by monitoring the behaviour of vertices in the corresponding sequence of graphs.

Monitoring the behaviour of every vertex in the graph is a challenging problem because each graph in the time sequence contains a large number of vertices resulting in a high-dimensional mathematical object. Spectral embedding methods provide an effective solution to the high dimensionality problem (Section 2.5). Although there are various spectral embedding-based change detection methods available (Section 2.4), these methods do not address sparsity and degree heterogeneity issues that are present in most real-world graphs. Consequently, changes involving only a few vertices, or changes involving low degree vertices, tend to be missed by these methods.

In this chapter, we propose a novel method called CDP (change detection using Procrustes analysis) to detect changes in vertex behaviour. In our method, we first obtain a

low dimensional embedding from the weighted adjacency matrix representing the graph at each time instant. Each embedded point characterizes the behaviour of a vertex in the graph at a given time instant. We use Procrustes analysis techniques to compare embeddings across time instants and calculate change scores for vertices. We evaluate the performance of CDP on several simulation experiments and the dynamic network for the Enron email dataset. Based on the results, we conclude that CDP efficiently and effectively identifies various vertex-based changes that are considered in our experiments.

This chapter is organized as follows. We first provide a brief overview of our overall change detection method in Section 3.1. In Section 3.2, we provide a detailed description of our change detection framework. Then we build our proposed algorithm step by step (Sections 3.2.3, 3.2.4 and 3.2.5) employing the methodologies discussed in Chapter 2. In Section 3.2.6, we summarize our change detection procedure and present the CDP algorithm. We evaluate the performance of CDP using simulation experiments (Section 3.3) and a real-world application (Section 3.4). In each experiment, the performance of CDP is compared with two other change detection approaches which are discussed in Section 3.3.5. Finally, we conclude the chapter by summarizing our findings in Section 3.5.

3.1 Brief Overview

In this chapter, we propose a novel method called CDP (change detection using Procrustes analysis) to detect vertex-based changes in a dynamic network. CDP employs a spectral approach to extract *features* for vertices at each time instant. A feature is a $d-$dimensional vector, that provides a representative summary of the behaviour of the corresponding vertex. Recall that we represent a dynamic network as a time sequence of undirected graphs, where each graph is then represented as a symmetric, weighted adjacency matrix. As we are dealing with large and sparse graphs, the weighted adjacency matrix is also large and noisy (a detailed discussion was provided in Section 2.5), making it hard to study the underlying structure. By applying spectral methods to the weighted adjacency matrix, we can embed the vertices into a d-dimensional Euclidean space that preserves the *closeness* (as defined in Section 2.5) between vertices in the original graph representation. The embedded points also highlight important vertex properties such as transitivity, homophily by attributes, and clustering, that are present in most real-world graphs (Section 2.3.1). In this chapter, we consider these embedded points as features for vertices characterizing vertex behaviour at each time instant. Vertices in sparse and heterogeneous graphs depict entities with different abilities to establish connections. It is difficult to achieve a good representation if we ignore sparseness and degree heterogeneity

when obtaining a low dimensional embedding. By employing ideas from spectral graph theory, combined with Amini et al. (2013)'s graph regularization technique (reviewed in Section 2.5.1), we formulate a strategy to effectively embed sparse and heterogeneous graphs into low dimensional Euclidean spaces. We also adapt Achlioptas and McSherry (2007)'s low-rank matrix approximation method (Section 2.5.1.6) to automatically estimate the proper embedding dimension.

In Section 2.6.2, we discussed how generalized orthogonal Procrustes analysis (GPA) methods can be used to calculate an average from a set of matrices after removing Euclidean similarity transformations. We adjust the standard GPA technique to extract *profile features* during the recent past time instants, and calculate change scores for vertices at each time instant. A profile feature, which is also a vector, represents the average behaviour of the vertex in the recent past time instants (previous w time instants). Our idea of applying Procrustes analysis techniques to compare embeddings for the purpose of change detection in dynamic networks is new and is inspired by Tang et al. (2012). Using a moving window approach, the change score calculation procedure is repeated over time to detect changes for all time instants.

Figure 3.1 provides an illustration of the overall CDP framework. In order to evaluate the performance of CDP, we apply it to both synthetic and real-world datasets. We compare our method with two baseline change detection methods that are also based on different spectral embedding procedures. The results show that CDP performs better than the others in various change scenarios considered.

3.2 Problem Framework

3.2.1 Notation and Terminology

Let $G^1, G^2, \ldots, G^{\mathcal{T}}$ be a sequence of graphs defined over time instants, $t = 1, 2, \ldots, \mathcal{T}$. Each G^t is a weighted and undirected graph with a fixed set of vertices, $V = \{v_1, \ldots v_n\}$. In our discussions, we also refer to v_i as vertex i. Define the edge set of graph, G^t, as E^t, where $|E^t| \leq n^2$, and E^t contains edge, $e_{i,j}$, if there is an edge between vertex i and vertex j. Each graph is represented by a symmetric weighted adjacency matrix, W^t, of dimension $n \times n$, where each element, $W^t_{i,j} \geq 0$. If $W^t_{i,j} = 0$, then the vertices i and j are not connected in G^t. The degree of each vertex i at time instant t is defined as

$$d^t_i = \sum_{j=1}^{n} W^t_{i,j}.$$

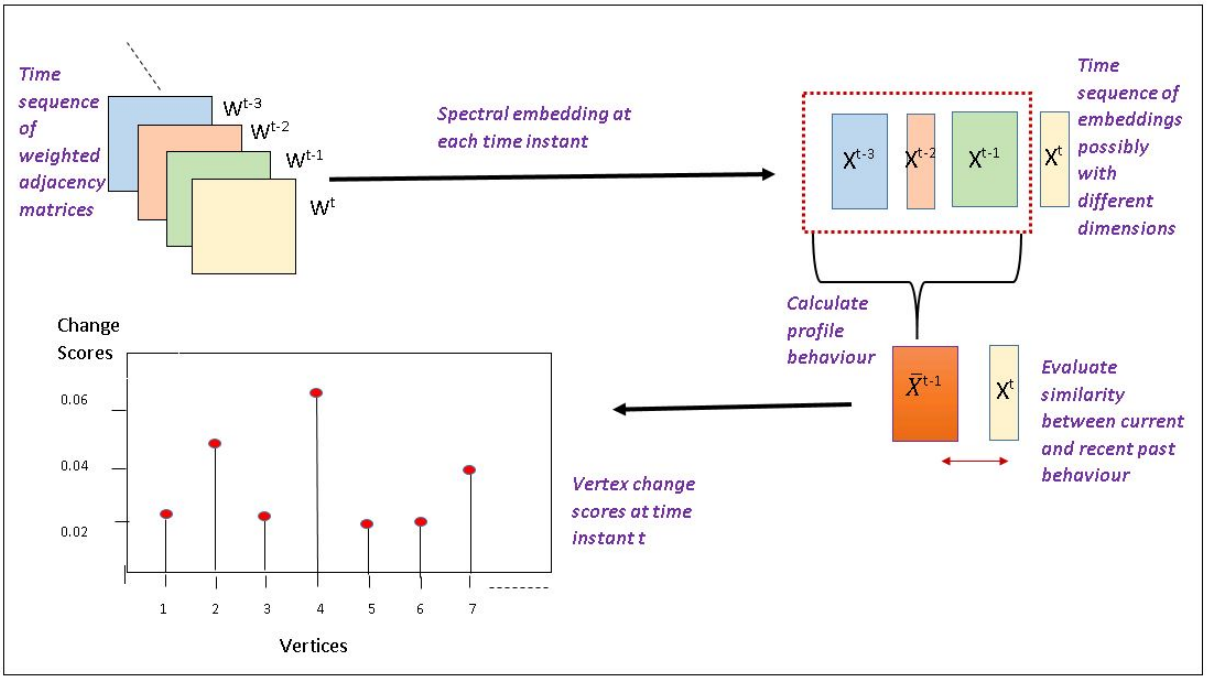


FIGURE 3.1: **Illustration of the overall CDP framework**. The time sequence of graphs is first represented as a time sequence of weighted adjacency matrices. At each time instant, we perform spectral embedding on the matrix and obtain an embedding where each row corresponds to a feature representing a vertex's behaviour. Next, we define a window of length, $w \in \mathbb{Z}_+$, over the previous w embeddings, and use GPA to obtain the profile embedding, where each row corresponds to a vertex's profile feature. The dissimilarity between the current embedding and the profile embedding is then obtained to compute the change scores of the vertices at the current time instant. The window is moved along all preceding time instants to calculate vertex change scores for the whole time period.

The degree matrix, D^t, is the diagonal matrix containing the vertex degrees, $d_1^t, \ldots, d_n^t$, on the diagonal. Let $\hat{\lambda}^t$ be the average vertex degree of graph, G^t, where $\hat{\lambda}^t = \frac{1}{n}\sum_i d_i^t$. From Amini et al. (2013), we define a network as sparse when $\hat{\lambda}^t < 5$.

3.2.2 Problem Statement

At each time instant t, our goal is to calculate a change score for each v_i in G^t, relative to the recent past behaviour. Our definition of the change score for v_i at time instant t is defined as follows.

Definition 3.1. The change score, z_i^t, for v_i at time instant t is

$$z_i^t = f(\bar{\mathbf{x}}_i^{t-1}, \mathbf{x}_i^t), \tag{3.1}$$

where $\mathbf{x}_i^t$ is the feature vector representing the behaviour of v_i at time instant t, $\bar{\mathbf{x}}_i^{t-1}$ is the profile feature vector representing the behaviour of v_i in the recent past time instants, and f is a dissimilarity function.

According to this definition, our overall change detection procedure can be summarized as follows.

1. Obtain a feature, $\mathbf{x}_i^t$, for v_i from each G^t, where $\mathbf{x}_i^t \in \mathbb{R}^{d^t}$ and $d^t \in \mathbb{Z}_+$.
2. Obtain a profile feature, $\bar{\mathbf{x}}_i^{t-1}$, for v_i from recent w past time instants, $G^{t-w}, \ldots, G^{t-1}$, where $\bar{\mathbf{x}}_i^{t-1} \in \mathbb{R}^{\bar{d}^{t-1}}$ and $\bar{d}^{t-1} \in \mathbb{Z}_+$.
3. Calculate the dissimilarity between $\mathbf{x}_i^t$ and $\bar{\mathbf{x}}_i^{t-1}$, and obtain the *change scores*, z_i^t, for $v_i \in V$ by using a suitable dissimilarity function f.

In Sections 3.2.3, 3.2.4, and 3.2.5, we discuss how these steps are implemented respectively.

3.2.3 Feature Extraction at Each Time Instant

In this section, we formulate our spectral embedding strategy for each G^t based on the methodology discussed in Section 2.5.1[1]. The *embedding* of a graph is an $n \times d$ matrix, where rows correspond to the $d-$dimensional embedded points for vertices. Our spectral embedding procedure consists of three main steps.

1. Pre-processing the weighted adjacency matrix, W, of G.
   As we consider weighted, heterogeneous graphs, some edges possess considerably higher weights than the other edges and can turn out to be very influential during the embedding process. These edges are called *dominant edges*. The elements of the corresponding weighted adjacency matrix, W, also show high variability. The presence of dominant edges may also hinder the detection of unusual edges that have lower weights, preventing the change from being detected. Applying a transformation on W, such as the logarithm, helps to mitigate this problem. After the log transformation, we scale each element, so that all elements in the resulting matrix are between zero and one. Scaling enables results obtained across views to be comparable in multi-view networks (we find this mostly useful when we extend our current method to multi-view networks in Chapter 4). Below we state our two preprocessing steps in more detail.

[1]Note that in this chapter, for discussions focused on one time instant, we drop the superscript t to simplify notation (for example, we use W instead of W^t to denote the matrix of G^t).

(a) Apply a log transformation to each element in W, and obtain $\ddot{W}$, where

$$\ddot{W}_{i,j} = \log_{10}(W_{i,j} + 1) \quad \forall i, j \in \{1, \ldots, n\}. \tag{3.2}$$

(b) Scale the elements of $\ddot{W}$ by its maximum element, and obtain $\grave{W}$, where

$$\grave{W}_{i,j} = \frac{\ddot{W}_{i,j}}{\max_{i,j}\{\ddot{W}_{i,j}\}}. \tag{3.3}$$

Note that the methodology discussed in this Chapter is also applicable to an unweighted graph, where the representation matrix is the binary adjacency matrix, A, with elements that are 0's or 1's. However, performing log transformation followed by scaling would make no difference, hence can be omitted in this case.

2. Obtaining a suitable representation matrix.
The mapping of edge weights into a suitable representation matrix is an essential task when using the embedded points to study the structure of the underlying graph (Skillicorn, 2007a). In sparse and heterogeneous graphs possessing power law degree distributions, the embeddings from the weighted adjacency matrix will only focus on vertices with the highest degrees, resulting in an inaccurate representation of the underlying connectivity structure (Mihail and Papadimitriou, 2002). To account for sparsity and degree heterogeneity, we construct the *regularized degree normalized weighted adjacency matrix*, M, as the representation matrix. Let the regularizer, τ, be

$$\tau = \frac{1}{4n^2} \sum_{i,j} \grave{W}_{i,j}. \tag{3.4}$$

Then M is given by

$$M = D_\tau^{-1/2} W_\tau D_\tau^{-1/2}, \tag{3.5}$$

where

$$W_\tau = \grave{W} + \tau \mathbf{1}\mathbf{1}^T, \tag{3.6}$$

where $\mathbf{1}$ is an n-dimensional column vector containing all ones, and D_τ is the degree matrix for W_τ. The regularization step (Equation 3.6) addresses sparseness by adding τ to each element in $\grave{W}$, while the degree normalization step (Equation 3.5) further adjusts for the irregularity in the degree distribution by dividing each element, $[W_\tau]_{i,j}$, by $\sqrt{[d_\tau]_i [d_\tau]_j}$. For a detailed theoretical justification on using M as the representation matrix to obtain an embedding, we refer the reader to Section 2.5.1.

3. Obtaining a low dimensional embedding from the representation matrix, M, using spectral decomposition.

From our discussion in Section 2.5.1, a low dimensional embedding, X, from the representation matrix, M can be seen as a solution to the optimization function,

$$\max_{X} \parallel X^T M X \parallel_F^2, \tag{3.7}$$

subject to $X^T X = I$, where $X \in \mathbb{R}^{n \times d}$ for $d \ll n$. From Section 2.5.1.6, X can be estimated by performing the singular value decomposition (SVD), $M = U\Sigma V^T$, and extracting d principal singular vectors. In order to determine d we employ Achlioptas and McSherry (2007)'s low-rank matrix approximation procedure. They propose to retain those singular vectors which capture the strongest structure in M based on the L_2 norm (Equation 2.12). A detailed description of their method was given in Section 2.5.1.6.3. In this section, we summarize our implementation of their method in Algorithm 1.

Algorithm 1 Optimal Low-Rank d Approximation

Input: (i) Symmetric matrix, M, with dimensions $n \times n$, where $\text{rank}(M) = r$, (ii) threshold, ϵ

Output: d

1: Compute SVD, $M = U\Sigma U^T$
2: Update $M = M - \sigma_1 \mathbf{u}_1 \mathbf{u}_1^T$
3: Compute SVD, $M = U\Sigma U^T$
4: Initialize k=1, $\rho = \inf$
5: **while** $\rho > \epsilon$ **and** $k \leq r$ **do**
6: $\quad \hat{M}_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{u}_j^T$
7: $\quad R_k = M - \hat{M}_k$
8: $\quad$ Calculate $\tilde{R}_k$ by randomly flipping the signs of elements in R_k such that, $\mathbb{P}\left[[\tilde{R}_k]_{i,j} = [R_k]_{i,j}\right] = \frac{1}{2}$ and $\mathbb{P}\left[[\tilde{R}_k]_{i,j} = -[R_k]_{i,j}\right] = \frac{1}{2}$
9: $\quad$ Update $\rho = \frac{|\|R_k\|_2 - \|\tilde{R}_k\|_2|}{\|R_k\|_F}$
10: $\quad$ Set $k = k + 1$
11: **end while**
12: **if** $k = r$ **then**
13: $\quad$ Set the converged value $d = r$
14: **else**
15: $\quad$ Set the converged value $d = k - 1$
16: **end if**

It is important to note that the regularization step (Equation 3.6) inserts edges between all disconnected components and creates a connected graph. For such a graph, the first principle singular vector, $\mathbf{u}_1$ (with corresponding singular value σ_1), of M, is a constant vector and therefore not useful for the embedding (Section 2.5.1). Thus, to obtain the embedding dimension, d, we initially remove the first reconstruction in step 2 of Algorithm 1. Hence, the output d returned by the

algorithm is the number of principal singular vectors that should be kept starting from the second principal singular vector onwards[2]. Once d is obtained, the low dimensional embedding, $X \in \mathbb{R}^{n \times d}$, is given by

$$X = [\mathbf{u}_2, \dots \mathbf{u}_{d+1}].$$

Each row vector, $\mathbf{x_i} \in \mathbb{R}^d$, is the feature for v_i at a given time instant.

Furthermore, an important input parameter for Algorithm 1 is the convergence threshold, ϵ. Since there is no definitive method for choosing ϵ discussed in Achlioptas and McSherry (2007), we conduct extensive experiments to select ϵ. Our results detailed in Appendix B, indicate that $\epsilon = 0.005$.

By following steps 1, 2, and 3 in Section 3.2.3, each graph, G^t, in the time sequence is represented as a low dimensional embedding, $X^t \in \mathbb{R}^{n \times d^t}$, where d^t is the embedding dimension returned by Algorithm 1. After following the three steps discussed in this section, the sequence of graphs, $G^1, \dots, G^{\mathcal{T}}$, is reduced to a sequence of low dimensional embeddings, $X^1, \dots, X^{\mathcal{T}}$.

3.2.4 Obtaining the Profile Features at Each Time Instant

After performing the steps stated in Section 3.2.3, we have a set of w embeddings from the w recent past time instants. From the *uniqueness* property of SVD (Section 2.5.1.6.1), the embedding obtained at each time instant is unique up to Euclidean similarity transformations such as scale, rotation and reflection. Thus, we cannot directly average the embeddings from the recent past time instants to obtain profile features. In Section 2.6.2, we discussed a technique called generalized orthogonal Procrustes analysis (GPA) that can be used to obtain an average from a set of matrices after adjusting for Euclidean similarity transformations. In this section, we show how we employ GPA to obtain an average embedding, $\overline{X}^{t-1}$, from the set of embeddings, $X^{t-w}, \dots, X^{t-1}$. We call $\overline{X}^{t-1}$, the *profile embedding* for time instant t. Let us first recall the GPA procedure.

From Section 2.6, the *pre-shape*, $\widetilde{X}$, of a matrix, $X \in \mathbb{R}^{n \times d}$, is defined as

$$\widetilde{X} = \frac{X_c}{\| X_c \|_F}, \tag{3.8}$$

where

$$X_c = CX, \tag{3.9}$$

[2] Achlioptas and McSherry (2007) show that the Frobenius norm of a matrix measures its *average linear trend* . Hence, the division by the Frobenius norm of R_k in step 9 of the algorithm provides a standardization to each ρ_k (Skillicorn, 2007a).

and the centering matrix, $C = I - \frac{1}{n}\mathbf{j}\mathbf{j}^T$. Here, I is an $n \times n$ identity matrix, and $\mathbf{j}$ is an $n-$dimensional vector of ones. Let $X_1, \ldots, X_w$ be w matrices, each of dimension $n \times d$. GPA involves the optimization of the least squares objective function

$$\min_{\Gamma,\mu} \sum_{i=1}^{w} \parallel \widetilde{X}_i\Gamma_i - \mu \parallel_F^2, \tag{3.10}$$

where $\Gamma_i \in \mathbb{R}^{d\times d}$ is the orthogonal rotation/reflection matrix corresponding to X_i, and $\widetilde{X}_i$ is the preshape corresponding to X_i as given in Equation 3.8. In Algorithm 2 we summarize our implementation of the iterative algorithm that solves the GPA objective function.

Algorithm 2 Generalized Procrustes Distance Calculation

Input: $X_1, \ldots, X_w \in \mathbb{R}^{n\times d}$, threshold ϵ
Output: $\hat{\mu}$, $\widehat{X}_i$ for $i = 1, \ldots, w$

Initialize $\mu_0 = X_1$, $\mathcal{D} = \inf$
while $\mathcal{D} > \epsilon$ **do**
  for $i \in \{1, \ldots, w\}$ **do**
    Calculate $\widetilde{X}_i = \frac{CX_i}{\|CX_i\|_F}$, where $C = I - \frac{1}{n}\mathbf{j}\mathbf{j}^T$
    Calculate SVD, $\mu_0^T\widetilde{X}_i = U\Sigma V^T$
    Calculate $\hat{\Gamma}_i = VU^T$
    Obtain $\hat{X}_i = \widetilde{X}_i\hat{\Gamma}_i$
  end for
  Calculate mean embedding, $\hat{\mu} = \frac{1}{w}\sum_{i=1}^{w}\widehat{X}_i$, using the aligned embeddings
  Update $\mathcal{D} = \parallel \mu_0 - \hat{\mu} \parallel_F^2$
  Update: $\mu_0 = \hat{\mu}$
end while

There is one limitation in applying GPA to the embeddings obtained at different time instants. GPA assumes that all matrices, $X^{t-w}, \ldots, X^{t-1}$, are of the same dimension, but the embeddings resulting from our methods discussed in Section 3.2.3, can be of different dimensions. We find two possible solutions to address this problem. Let $d_{\max} = \max\{d^{t-w}, \ldots, d^{t-1}\}$.

1. For any X^t with $d^t < d_{\max}$, append columns of zeros to X^t to make it of size $n \times d_{\max}$.

2. For any X^t with $d^t > d_{\max}$, truncate the additional columns of X^t to make it of size $n \times d_{\max}$.

Truncating extra dimensions causes us to drop singular vectors that may describe important structure of the graph. Appending columns of zeros does not cause loss of information, and is thus preferred. We have also conducted experiments to check that this results in better change detection performance. Hence, whenever the dimensions

of the embeddings to be compared are different from each other, we append the low dimensional embedding with columns of zeros before fitting the generalized Procrustes model.

Therefore, the profile embedding, $\overline{X}^{t-1}$, is calculated as follows:

1. Let $d_{\max} = \max\{d^{t-w} \ldots, d^{t-1}\}$. Append $d_{\max} - d^t$ columns of zeros to each X^t and obtain X^t_{padded}.
2. Perform the generalized Procrustes analysis procedure and estimate the mean embedding, $\overline{X}^{t-1} = \hat{\mu}$. To do this, we input $X^{t-w}_{\text{padded}}, \ldots, X^{t-1}_{\text{padded}}$ into Algorithm 2, and estimate the mean embedding, $\hat{\mu}$.

At each time instant t, the n rows of $\overline{X}^{t-1}$ give the profile features for the n vertices in the graph.

3.2.5 Change Score Calculation

After applying the methods discussed in Sections 3.2.3 and 3.2.4, at each time instant t, we end up with the profile embedding, $\overline{X}^{t-1} \in \mathbb{R}^{n \times \bar{d}^{t-1}}$, and current embedding, $X^t \in \mathbb{R}^{n \times d^t}$. Vertex change scores are calculated by computing the dissimilarity between X^t and $\overline{X}^{t-1}$. As described in Section 2.6, Procrustes analysis can be used to compare two matrices after adjusting for Euclidean similarity transformations. From Section 3.2.4, when $\bar{d}^{t-1} \neq d^t$, we append columns of zeros to the lower dimensional embedding. Thus, the change score, z_i^t, for vertex i at time instant t is calculated as follows:

1. Let $d_{\max} = \max\{d^t, \bar{d}^{t-1}\}$. Append $d_{\max} - d^t$ and $d_{\max} - \bar{d}^{t-1}$ columns of zeros to X^t and $\overline{X}^{t-1}$, respectively and obtain $X^t_{\text{padded}} \in \mathbb{R}^{n \times d_{\max}}$ and $\overline{X}^{t-1}_{\text{padded}} \in \mathbb{R}^{n \times d_{\max}}$.
2. Perform GPA using Algorithm 2 and obtain the transformed embeddings, $\widehat{X}^t$ and $\widehat{\overline{X}}^{t-1}$, and the average of the transformed embeddings, $\hat{\mu}^t$.
3. For each vertex i, calculate the change score

$$z_i^t = \frac{\| \widehat{X}^t_{i,\cdot} - \widehat{\overline{X}}^{t-1}_{i,\cdot} \|_F^2}{\| \hat{\mu}^t \|_F}. \tag{3.11}$$

3.2.6 Proposed Algorithm - CDP (Change Detection using Procrustes Method)

We have now constructed the three main steps of our change detection procedure. These include, at each time instant, extracting features for vertices through graph embedding

(Section 3.2.3), calculating profile features for vertices by applying GPA on the recent past embeddings (Section 3.2.4), and finally calculating change scores for vertices through generalized Procrustes distance calculation between current and profile embeddings (Section 3.2.5). The steps are listed in Algorithm 3.

Algorithm 3 Change Detection using Procrustes Analysis - CDP

Input: (i) Time sequence of symmetric, weighted adjacency matrices, $W^1, W^2, \dots W^{\mathcal{T}}$, where each W^t has dimension $n \times n$ (ii) window size, w

Output: Time sequence of vertex change scores, $\mathbf{z}^1, \mathbf{z}^2, \dots \mathbf{z}^{\mathcal{T}}$. Each $\mathbf{z}^t$ is a vector of dimension n

for $t = 1$ **to** $\mathcal{T}$ **do**
  Update: $W^t = \log_{10}(W^t + \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)$
  Update: $W^t = \frac{W^t}{\max_{i,j}\{W^t_{i,j}\}}$
  Calculate $\tau^t = \frac{1}{4n^2}\sum_{i,j} W^t_{i,j}$
  Update: $W^t_\tau = W^t + \tau^t \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T$,
  Calculate $D^t_\tau \in \mathbb{R}^{n\times n}$, where $[D^t_\tau]_{i,i} = \sum_{j=1}^n [W^t_\tau]_{i,j}$
  Calculate $M^t = (D^t_\tau)^{-1/2} W^t_\tau (D^t_\tau)^{-1/2}$
  Input M^t to Algorithm 1 and estimate d^t
  Perform SVD: $M^t = U\Sigma U^T$
  Obtain the low dimensional embedding $X_t \in \mathbb{R}^{n\times d^t}$, where $X^t = [\mathbf{u}_2, \dots \mathbf{u}_{d^t+1}]$
end for
for $t = w + 1$ **to** $\mathcal{T}$ **do**
  Let $d_{\max_1} = \max\{d^{t-w}, \dots, d^{t-1}\}$
  for $t' \in \{t - w, \dots, t - 1\}$ **do**
    if $d^{t'} < d_{\max_1}$ **then**
      append $d_{\max_1} - d^{t'}$ columns of zeros to $X^{t'}$
    end if
  end for
  Input $X^{t-w}, \dots, X^{t-1}$ into Algorithm 2, and estimate the profile embedding $\overline{X}^{t-1}$
  Let $d_{\max_2} = \max\{d_{\max_1}, d^t\}$
  if $d_{\max_1} < d_{\max_2}$ **then**
    append $d_{\max_2} - d_{\max_1}$ columns of zeros to $\overline{X}^{t-1}$
  end if
  if $d^t < d_{\max_2}$ **then**
    append $d_{\max_2} - d^t$ columns of zeros to X^t
  end if
  Align X^t and $\overline{X}^{t-1}$ with each other using Algorithm 2, and obtain the adjusted embeddings, $\widehat{X}^t$ and $\widehat{\overline{X}}^{t-1}$, and the mean $\hat{\mu}^t$
  Calculate vertex change scores, $z^t_i = \frac{\|\widehat{X}^t_{i,.} - \widehat{\overline{X}}^{t-1}_{i,.}\|_F^2}{\|\hat{\mu}^t\|_F}$
end for

After describing our algorithm, we evaluate its performance by conducting experiments on simulated dynamic networks and a real-world dataset.

3.3 Simulation Experiments

We conduct experiments with different synthetic datasets that mimic several real-world change scenarios. Within each scenario, the behaviour of a subset of vertices change.

3.3.1 Overall Setting

For each change scenario we generate a time sequence of symmetric weighted adjacency matrices, $W^1, W^2, \ldots, W^{\mathcal{T}}$, to represent a time sequence of weighted graphs. We assume that the edges of the graphs have distribution F_0 and when a change occurs the distribution becomes F_1. We consider two types of changes.

1. Change occurs at a given time instant: *change-point*. A change point is injected to the time sequence of graphs by defining the edge distribution as

$$W^t \sim \begin{cases} F_1 & \text{if } t = t^*, \\ F_0 & \text{otherwise }, \end{cases} \tag{3.12}$$

   for $w < t^* \leq \mathcal{T}$.

2. Change occurs at a time instant, and persists for some time period: *change-interval*. A change-interval is generated by defining the edge distribution as

$$W^t \sim \begin{cases} F_1 & \text{if } t_1^* \leq t \leq t_2^*, \\ F_0 & \text{otherwise }, \end{cases} \tag{3.13}$$

   for $w < t_1^* < t_2^* \leq \mathcal{T}$.

In Section 2.3, we discussed many random graph models that can be used to generate edges for a graph. In the next section, we discuss the model that is used to generate graphs for experiments in this chapter.

3.3.2 Random Graph Model Used for Synthetic Network Generation

The degree corrected stochastic block model (DCSBM) (Section 2.3.3.1) is a commonly used model because it can closely mimic the community structure of real-world networks. In our simulation experiments, we employ the DCSBM to define the probability distribution of the edges of a graph. By adjusting the model parameters, we obtain a wide variety of edge distributions.

Let $c_i \in \{1, 2, \ldots, k\}$ denote the block membership of vertex i. Then the vector, $\mathbf{c} \in \{1, 2, \ldots, k\}^n$, of dimension n denotes the block memberships of the n vertices in the graph. In terms of the weighted adjacency matrix, W, its distribution under the DCSBM is given by

$$\mathbb{P}[W|\boldsymbol{\theta}, \psi, \mathbf{c}] = \prod_{i \neq j} \frac{(\theta_i \theta_j \psi_{c_i,c_j})^{W_{i,j}}}{W_{i,j}!} \exp(-\theta_i \theta_j \psi_{c_i,c_j}), \tag{3.14}$$

where ψ_{c_i,c_j} is the expected number of edges between a vertex in block c_i and a vertex in block c_j, and $\boldsymbol{\theta}$ is an n-dimensional vector of degree parameters. Note that the previously defined probability, B_{c_i,c_j}, in Section 2.3.3.1 is replaced by the expected number, ψ_{c_i,c_j}, here. Since we generate large and sparse graphs, where B_{c_i,c_j} and ψ_{c_i,c_j} are equal in the limit as $n \to \infty$, the difference between model 2.23 and model 3.14 is inconsequential (Karrer and Newman, 2011). Each element, $W_{i,j}$, is a Poisson random variable with mean $\theta_i \theta_j \psi_{c_i,c_j}$.

In order to mimic the degree distribution of real-world graphs, the vector, $\boldsymbol{\theta}$, is generated from a power-law distribution (Clauset et al., 2009) defined as

$$\mathbb{P}(\boldsymbol{\theta}|\theta_{\min}, \beta) = \prod_{i=1}^{n} \frac{\beta - 1}{\theta_{\min}} \left(\frac{\theta_i}{\theta_{\min}}\right)^{-\beta},$$

where $\theta_{\min}$ is the lower bound of the support of θ_i, β is the shape parameter. The θ_i's are normalized to sum to one for vertices in the same block, i.e., $\sum_i \theta_i \delta_{c_i,r} = 1$ (where $\delta_{c_i,r} = 1$ if vertex i belongs to block r).

To specify what ψ is, let the block probability matrix, $B \in [0, 1]^{k \times k}$, be defined as in Section 2.3.3.1. Using $\mathbf{c}$, we can obtain $\mathbf{g} \in \mathbb{R}^{k \times 1}$, where each element, $g_r = \sum_{i=1}^{n} \delta_{c_i,r}$, denotes the number of vertices in block r. Using B and $\mathbf{g}$ we can calculate the expected number of edges, $\psi_{r,s}$, between a vertex in block r and a vertex in block s giving

$$\psi_{r,s} = B_{r,s} g_r g_s.$$

We select B to have the form

$$B = \lambda B^{\text{planted}} + (1 - \lambda) B^{\text{random}}, \tag{3.15}$$

where $\lambda \in [0, 1]$. For example, for a graph with three blocks, B^{planted} can take the form,

$$B^{\text{planted}} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}, \tag{3.16}$$

where $\alpha, \beta, \gamma \in [0, 1]$ give the intra-block probabilities. B^{random} is given by

$$B^{\text{random}} = \nu \mathbf{1}_k \mathbf{1}_k^T, \tag{3.17}$$

where $\mathbf{1}_k$ is the $k \times 1$ vector of ones, and $\nu \in [0, 1]$. ν can be regarded primarily as an inter-block probability. Thus, by varying λ, we can vary the level of noise in the generated graphs, which makes it more difficult to identify the blocks.

The distributions, F_0 and F_1, for the edges are both given by the DCSBM with different sets of parameter values. Each set of parameter values is chosen to mimic real-world change scenarios involving vertices. In Table 3.1, we summarize the parameter settings of different DCSBM models used to generate graphs in our experiments.

3.3.3 Change Scenarios

In Section 2.4, we reviewed numerous change scenarios studied in previous research. Based on these ideas, we come up with the following change scenarios to evaluate our change detection method.

1. Change in block membership - *group-change*.
   A set of vertices in a block change their block (group) membership.
2. Change in block Structure (also discussed in Section 2.4.4.1),
   (a) *split* - a block in the graph splits into two blocks,
   (b) *merge* - the reverse of split: two blocks join together and form one block,
   (c) *form* - a high increase in connections in a block that was previously sparse,
   (d) *fragment* - the reverse of form: a dense block becomes sparse.
3. Change in degree,
   (a) Heterogeneous degrees to homogeneous degrees - *hetero-to-homo*.
   The degree parameters of a block of vertices in the graph change from heterogeneous to homogeneous.

   (b) Homogeneous degrees to heterogeneous degrees - *homo-to-hetero*.
   The reverse of hetero-to-homo: the degree parameters of a block of vertices change from homogeneous to heterogeneous.

4. Change in connectivity patterns:

   (a) Clear block structure to complex structure - *simple-to-complex*.
   Two blocks add inter-block edges, disrupting the clear block structure in the graph.

   (b) Complex block structure to clear block structure - *complex-to-simple*.
   The reverse of simple-to-complex: most inter-block edges between two blocks vanish, resulting in a graph with a clear block structure.

In Table 3.2, we give a detailed description of how we mimic these change scenarios through transitions of the underlying generative models. Each scenario corresponds to changes in the connectivity patterns of a subset of vertices in the graph. For each scenario, we visualize an example of W's generated from the models corresponding to F_0 and F_1.

For each change scenario, we generate a sequence of 30 graphs, that is, we set $\mathcal{T} = 30$. The parameters for the two types of changes defined in Section 3.3.1 are as follows.

1. change-point (Equation 3.12): $t^* = 21$,
2. change-interval (Equation 3.13): $t_1^* = 21, t_2^* = 30$.

We use windows of sizes $1, 5$, and 10, and calculate change scores for all vertices. We repeat this 100 times, and calculate our performance measures.

TABLE 3.1: Parameter settings of different models with fixed parameters, $n = 900, \lambda = 0.8$, B^{planted} with $\alpha = 0.01, \beta = 0.02, \gamma = 0.03$, and B^{random} with $\nu = 0.0025$.

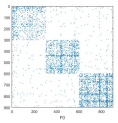
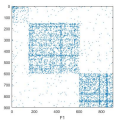
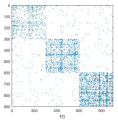
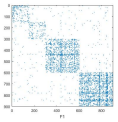
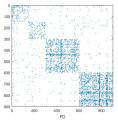
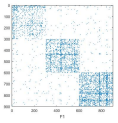
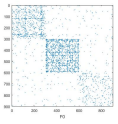
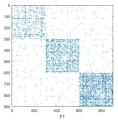
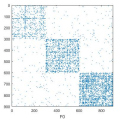
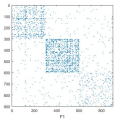
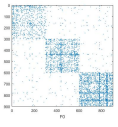
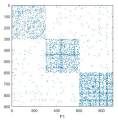
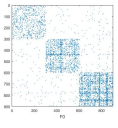
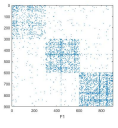
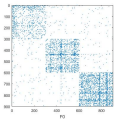
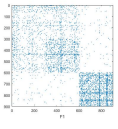
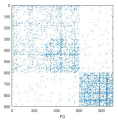
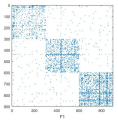
Model[a]	B^{planted}	Distribution of θ	$\mathbf{g}$	k
$\mathcal{M}1$	$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$	$\boldsymbol{\theta} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$	$[300, 300, 300]$	3
$\mathcal{M}2$	$\begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & \gamma \end{bmatrix}$	$\boldsymbol{\theta} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$	$[150, 150, 300, 300]$	4
$\mathcal{M}3$	$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 0.1(\gamma) \end{bmatrix}$	$\boldsymbol{\theta} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$	$[300, 300, 300]$	3
$\mathcal{M}4$	$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$	$\boldsymbol{\theta} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$	$[150, 450, 300]$	3
$\mathcal{M}5$	$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$	$\boldsymbol{\theta}_{V_c} = \mathbf{c}$ [b] $\boldsymbol{\theta}_{\bar{V}_c} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$ [c]	$[300, 300, 300]$	3
$\mathcal{M}6$	$\begin{bmatrix} 0.5\alpha & 0.5\alpha & 0 \\ 0.5\alpha & \beta - 0.5\alpha & 0 \\ 0 & 0 & \gamma \end{bmatrix}$	$\boldsymbol{\theta} \sim \mathbb{P}(\boldsymbol{\theta}\|1, 2.5)$	$[300, 300, 300]$	3

[a] Different values of λ were tested ($\lambda = 0, 0.1, 0.2, \ldots, 1$), but the same λ value is used for the pair of models involved in a given change scenario.

[b] $\boldsymbol{\theta}_{V_c}$ is a vector of degree parameters of the set of vertices, $V_c = \{v_1, v_2, \ldots, v_{300}\}$, and $\mathbf{c}$ denotes a positive vector of constants.

[c] $\boldsymbol{\theta}_{\bar{V}_c}$ is a vector of degree parameters of the set of vertices, $\bar{V}_c = \{v_{301}, v_{302}, \ldots, v_{900}\}$.

TABLE 3.2: **Illustration of change scenarios**. Each scenario corresponds to a change in the connectivity patterns of a subset of vertices in the DCSBM graph and is visualized using the pixel-plots of the adjacency matrices generated.

No.	Change Scenario	F_0	F_1	Changed Vertices
1	group-change	$\mathcal{M}1$ 	$\mathcal{M}4$ 	$\{1 \dots 600\}$
2	split	$\mathcal{M}1$ 	$\mathcal{M}2$ 	$\{1 \dots 300\}$
3	merge	$\mathcal{M}2$ 	$\mathcal{M}1$ 	$\{1 \dots 300\}$
4	form	$\mathcal{M}3$ 	$\mathcal{M}1$ 	$\{601 \dots 900\}$
5	fragment	$\mathcal{M}1$ 	$\mathcal{M}3$ 	$\{601 \dots 900\}$
6	hetero-to-homo	$\mathcal{M}1$ 	$\mathcal{M}5$ 	$\{1 \dots 300\}$
7	homo-to-hetero	$\mathcal{M}5$ 	$\mathcal{M}1$ 	$\{1 \dots 300\}$
8	simple-to-complex	$\mathcal{M}1$ 	$\mathcal{M}6$ 	$\{1 \dots 600\}$
9	complex-to-simple	$\mathcal{M}6$ 	$\mathcal{M}1$ 	$\{1 \dots 600\}$

3.3.4 Performance Measure

Since our goal is to detect vertices that have changed their behaviour with respect to the recent past, we measure the performance of CDP with respect to the ability of the change scores produced to discriminate between changed and unchanged vertices. Each change scenario discussed in Section 3.3.3 involves a set of vertices, V_c, changing their behaviour. Let $|V_c| = n_c$. If our method performs well, the change scores for vertices in V_c should be higher than the change scores for the rest of the vertices in $V_{\bar{c}}$, especially at the time instant corresponding to a change. Note that $|V_{\bar{c}}| = n_{\bar{c}}$, $V_c \cup V_{\bar{c}} = V$, and $n_c + n_{\bar{c}} = n$.

Let us consider a time sequence of vertex change scores, $\mathbf{z}^1, \mathbf{z}^2, \ldots \mathbf{z}^{30}$, where each $\mathbf{z}^t$ is a vector of length n obtained from a single simulation run of a change scenario. Let $\tilde{\mathbf{z}}^t$ be the $n_c \times 1$ vector of change scores obtained for V_c, and let $\bar{\mathbf{z}}^t$ be the $n_{\bar{c}} \times 1$ vector of change scores obtained for $V_{\bar{c}}$. We use a sampling procedure to estimate

$$\phi^t = \mathbb{P}[\tilde{z}_i^t > \bar{z}_j^t],$$

which is the probability that vertex, i, in V_c has a higher change score than vertex, j, in $V_{\bar{c}}$. We separately sample (with replacement) a vector of N elements, $\hat{\tilde{\mathbf{z}}}^t$, from $\tilde{\mathbf{z}}^t$ and a vector of N elements, $\hat{\bar{\mathbf{z}}}^t$, from $\bar{\mathbf{z}}^t$; then ϕ^t is calculated by counting the proportion of entries in $\hat{\tilde{\mathbf{z}}}^t$ that are larger than the corresponding entries in $\hat{\bar{\mathbf{z}}}^t$ as

$$\phi^t \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{\hat{\tilde{z}}_i^t > \hat{\bar{z}}_i^t},$$

where $\delta_{\hat{\tilde{z}}_i^t > \hat{\bar{z}}_i^t}$ is one if $\hat{\tilde{z}}_i^t > \hat{\bar{z}}_i^t$ and zero otherwise. In our experiments we use $N = 100000$.

A proportion greater than 0.5 indicates a higher chance of a change score for a vertex in V_c being greater than a change score for a vertex in $V_{\bar{c}}$. By repeating this for all 100 simulation runs, we obtain a 100×1 vector of probabilities, $\boldsymbol{\phi}^t$. If all elements of $\boldsymbol{\phi}^t$ are greater than 0.5 and closer to one at a changed time instant, good change detection performance is indicated.

In this chapter, instead of directly using ϕ_i^t, we use the log odds

$$\eta_i^t = \log\left(\frac{\phi_i^t}{1 - \phi_i^t}\right), \tag{3.18}$$

which measures the odds that a vertex in V_c has higher change scores than a vertex in $V_{\bar{c}}$. When a change occurs, we expect the values of $\boldsymbol{\eta}^t$ to lie above zero and be strongly positive.

After calculating η_i^t, we further calculate the log odds ratio between time instants t and $t-1$ which gives

$$\bar{\eta}_i^t = \log\left(\frac{\phi_i^t/(1-\phi_i^t)}{\phi_i^{t-1}/(1-\phi_i^{t-1})}\right). \tag{3.19}$$

In our experiments we calculate both $\boldsymbol{\eta}^t$ and $\bar{\boldsymbol{\eta}}^t$ to measure detection performance.

3.3.5 Comparison Methods

We compare our CDP algorithm with two baseline methods.

1. **ACT**

   This is the *activity* (ACT) vector-based change detection algorithm developed by Idé and Kashima (2004) which we discussed thoroughly in Section 2.4.1. Recall that Idé and Kashima (2004) employ a spectral embedding procedure, and represent a time sequence of graphs as a time sequence of activity vectors, $\mathbf{u}^t$, for $t \in \{1, 2, \ldots, \mathcal{T}\}$. A profile vector, $\mathbf{r}^{t-1}$, is calculated from recent past w activity vectors. The change score, z_i^t, can be calculated as

   $$z_i^t = |r_i^{t-1} - u_i^t|, \tag{3.20}$$

   where $|.|$ denotes absolute value. The elements of the activity vector, $\mathbf{u}^t$, denote the eigenvector centrality scores of the vertices in the graph. Idé and Kashima (2004) developed ACT to perform change detection in a time sequence of dense graphs. However in the majority of the applications we encounter, the graph obtained at each time instant is sparse and heterogeneous. As discussed in Section 3.1, such a graph consists of vertices with very high degree (*hubs*) as well as very low degree (sometimes zero; resulting in disconnected vertices in the graph). According to Martin et al. (2014), eigenvector centrality is a poor performance measure of centrality of vertices in sparse graphs. They show that the centrality scores are concentrated only on hubs and fail to capture the centrality of lower degree vertices. While this situation might be useful for some applications, for our current requirement of detecting changes in the behaviour of all vertices in the graph, it is inadmissible. Thus, we find Idé and Kashima (2004)'s approach cannot be generalized to most real-world graphs.

2. **ACTM**

   We make a slight improvement to Idé and Kashima's profile vector calculation step, and call this method the *modified activity* (ACTM) vector-based algorithm. Recall that Idé and Kashima represent the recent past behaviour using the profile vector, $\mathbf{r}^{t-1}$. However, $\mathbf{r}^{t-1}$ is only the first vector, $\mathbf{r}_1$, from the w singular

vectors, $[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_w]$, resulting from the SVD of the $n \times w$ matrix of activity vectors, $[\mathbf{u}^{t-1}, \ldots, \mathbf{u}^{t-w}]$, representing the recent past. The w left singular vectors, $[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_w]$, define an orthonormal basis for the subspace defined by the w activity vectors, $[\mathbf{u}^{t-w}, \ldots, \mathbf{u}^{t-1}]$. Selecting only the first vector, $\mathbf{r}_1$, might cause us to loose information. Hence, a more representative profile vector can be obtained by projecting $\mathbf{u}^t$ onto the w dimensional orthonormal subspace defined by $[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_w]$, where

$$\bar{\mathbf{r}}^{t-1} = \left(\mathbf{r}_1 \cdot \mathbf{u}^t\right) \mathbf{r}_1 + \ldots \left(\mathbf{r}_w \cdot \mathbf{u}^t\right) \mathbf{r}_w. \tag{3.21}$$

The profile vector, $\bar{\mathbf{r}}^{t-1}$ is also the best approximation to $\mathbf{u}^t$ in the subspace spanned by $[\mathbf{u}^{t-w}, \ldots, \mathbf{u}^{t-1}]$ (Poole, 2014). The error vector, $\mathbf{e}^t = \bar{\mathbf{r}}^{t-1} - \mathbf{u}^t$, gives an indication of the deviation of $\mathbf{u}^t$ from its recent past. Thus, the change score, z_i^t, is

$$z_i^t = |\bar{r}_i^{t-1} - u_i^t|. \tag{3.22}$$

3.3.6 Results

For each change scenario discussed in Section 3.3.3, we first calculate the performance measure $\boldsymbol{\eta}^t$ for several time instants before and after $t = 21$ for CDP, ACT, and ACTM for both change-point and change-interval. In Figure 3.2, we show the corresponding results for group-change with $w = 5$ for $t = 17, 18, \ldots, 30$. Let us first discuss the results of CDP. For all time instants, before $t = 21$, the $\boldsymbol{\eta}^t$'s are centred at a given level. All graphs generated before $t = 21$ are from the same model $\mathcal{M}1$. Since there is no model change, the odds of each $\mathbf{z}_{n_c}^t$ being greater than $\mathbf{z}_{n_{\bar{c}}}^t$ are similar during these time instants. At $t = 21$, the generative model changes to $\mathcal{M}4$, and we see a clear increase of $\boldsymbol{\eta}^{21}$ compared to $\boldsymbol{\eta}^{20}$. This shows that there is a clear increase in $\mathbf{z}_{n_c}^{21}$. From $t = 22$ onwards, we observe different patterns for change-point and change-interval.

- Change-point: the generative model returns to $\mathcal{M}1$ at $t = 22$ and persists for all time instants, $t = 23, \ldots, 30$.
  1. There is a big decrease in $\boldsymbol{\eta}^{22}$ compared to $\boldsymbol{\eta}^{21}$. Our window is $w = 5$. Inside the window, there are four graphs generated from $\mathcal{M}1$, and one graph generated from $\mathcal{M}4$. Thus, unlike at $t = 21$, there is less change compared to the recent past. Thus, $\boldsymbol{\eta}^{22}$ is less than $\boldsymbol{\eta}^{21}$.
  2. For $t = 22, 23, 24, 25, 26$, the window contains four graphs generated from $\mathcal{M}1$, and one graph generated from $\mathcal{M}4$. Thus, the change occurring in these time instants is similar. So, the $\boldsymbol{\eta}_t$'s are generally centred at the same level.

3. At $t = 27$, the window contains graphs generated purely from $\mathcal{M}1$, and the comparison is also done with a graph generated from $\mathcal{M}1$. So the change involving the set of vertices, V_c, at $t = 27$ is less than the change at $t = 26$. Hence, $\boldsymbol{\eta}^{27}$ decreases.
4. For $t = 27, 28, \ldots$, the window contains graphs generated purely from $\mathcal{M}1$, and the comparison is also done with a graph generated from $\mathcal{M}1$. Thus, the change occurring in these time instants is similar. So, the $\boldsymbol{\eta}_t$'s are generally centred at the same level.

- Change-interval: the generative model is $\mathcal{M}4$ for time instants, $t = 22, \ldots, 30$.
  1. There is a decrease in $\boldsymbol{\eta}^{22}$ compared to $\boldsymbol{\eta}^{21}$. Inside the window there are four graphs generated from $\mathcal{M}1$, and one graph generated from $\mathcal{M}4$. So there is less change involving the set of vertices, V_c, compared to their change at $t = 21$. Hence, $\boldsymbol{\eta}^{22}$ is less than $\boldsymbol{\eta}^{21}$.
  2. For all time instants, $t = 23, 24, 25$, the change becomes less and less as the window (recent past) contains more time instants which are similar to the current time instant. So $\mathbf{z}^t_{n_c}$ decreases with time, causing $\boldsymbol{\eta}^t$ to decrease accordingly.
  3. At $t = 26$, the window contains graphs generated purely from $\mathcal{M}4$, and the comparison is also done with a graph generated from $\mathcal{M}4$. So the change is less compared to the change at $t = 25$. Thus, $\boldsymbol{\eta}^{26}$ is less than $\boldsymbol{\eta}^{25}$.
  4. For $t = 26, 27, \ldots$, the window contains graphs generated purely from $\mathcal{M}4$, and the comparison is also done with a graph generated from $\mathcal{M}4$. Thus, the change occurring in these time instants is similar. So, the $\boldsymbol{\eta}_t$'s are generally centred at the same level.

For change detection methods ACT and ACTM, $\boldsymbol{\eta}^t$'s are wider. Furthermore, the bulk of $\boldsymbol{\eta}^t$ lies below zero for all time instants. Thus, although we see an increase in the $\boldsymbol{\eta}^{21}$ intervals for ACT and ACTM, these methods do not perform well in detecting the change.

Note that the graphs generated at each time instant are independent samples from a given generative model (F_0 or F_1). Thus, within the same generative model, edge weights can change from one time instant to another, also causing the connectivity patterns of vertices to change. For example, in Figure 3.2 (Top), we observe that the $\boldsymbol{\eta}^t$'s are centred at a positive level even within the generative model, $\mathcal{M}1$. This shows that the set of vertices, V_c, for the group-change scenario (Table 3.2) undergo a higher change in their connectivity patterns for independent graph realizations under $\mathcal{M}1$. However, when calculating our performance measure, the set of vertices, V_c, does not necessarily contain those vertices whose connectivity patterns have changed between independent

realizations from a given generative model. For example, in Figure 3.2 (Bottom), we observe that $\boldsymbol{\eta}^t$'s are centred at a negative level within generative model, $\mathcal{M}4$. This shows that the vertices in the set, $V_{\bar{c}}$, are the ones that change more during independent graph realizations under $\mathcal{M}4$. Despite these changes occurring in connectivity patterns within a given generative model, our interest lies in detecting a change during model transitions. At $t = 21$, we expect $\boldsymbol{\eta}^{21}$ to be larger than the $\boldsymbol{\eta}^t$'s observed for time instants corresponding to the same model. Thus, in order to clearly observe this, we calculate the performance measure $\bar{\boldsymbol{\eta}}^t$ (Equation 3.19). In Figure 3.3, we plot $\bar{\boldsymbol{\eta}}^t$ for CDP, ACT, and ACTM with $w = 5$ for the group-change scenario for $t = 17, 18, \ldots, 30$. We observe that $\bar{\boldsymbol{\eta}}^t$ provides a clearer picture than $\boldsymbol{\eta}^t$ on a method's ability to detect change caused by model transitions. For the rest of the scenarios in this chapter, we only plot $\bar{\boldsymbol{\eta}}^t$ over time and compare the performance measure, $\boldsymbol{\eta}^t$ (Equation 3.18), for CDP, ACT, and ACTM for all window sizes only at the time instant corresponding to a change, i.e., we only compare $\boldsymbol{\eta}^{21}$. For the interested reader, results showing the variation of $\boldsymbol{\eta}^t$ over time are included in Appendix C.

We compare the $\boldsymbol{\eta}^{21}$'s for only those change scores obtained on change-point scenarios since it is sufficient to calculate $\boldsymbol{\eta}^{21}$ for either change-point or change-interval as both involve similar changes when considering only $t = 21$. If $\boldsymbol{\eta}^{21}$ is positive, then this indicates that the vertices in the set, V_c, have higher change scores compared to the rest of the vertices in $V_{\bar{c}}$, at the time instant of change ($t = 21$). Figure 3.4 shows $\boldsymbol{\eta}^{21}$ returned by CDP, ACT, and ACTM for the group-change scenario using various window sizes, $w = 1, 5, 10$. The $\boldsymbol{\eta}^{21}$ returned by CDP for all window sizes are clearly positive. For $\boldsymbol{\eta}^{21}$ returned by ACT and ACTM, we see the bulk of the interval lying below zero for all window sizes, showing failure in detection for those methods. We also observe $\boldsymbol{\eta}^{21}$ for the other change scenarios, split (Figure 3.5), merge (Figure 3.7), form (Figure 3.9), fragment (Figure 3.11), hetero-to-homo (Figure 3.13), homo-to-hetero (Figure 3.15), simple-to-complex (Figure 3.17), and complex-to-simple (Figure 3.19). Our results show that CDP successfully detects the change in all the scenarios considered. ACT shows failure in detection for all change scenarios except form and fragment, while ACTM shows failure in detection for all change scenarios except form.

We further observe $\bar{\boldsymbol{\eta}}^t$ for split (Figure 3.6), merge (Figure 3.8), form (Figure 3.10), fragment (Figure 3.12), hetero-to-homo (Figure 3.14), homo-to-hetero (Figure 3.16), simple-to-complex (Figure 3.18), and complex-to-simple (Figure 3.20). CDP shows a clear detection at $t = 21$ for change scenarios form, fragment, hetero-to-homo, homo-to-hetero, simple-to-complex, and complex-to-simple. In the case of split and merge, we observe an increase at $\bar{\boldsymbol{\eta}}^{21}$, with the intervals being wide. ACT and ACTM do not show a clear increase at $\bar{\boldsymbol{\eta}}^{21}$ for split, merge, simple-to-complex, and complex-to-simple cases. For homo-to-hetero and hetero-to-homo, we observe a slight increase in $\bar{\boldsymbol{\eta}}^{21}$ for ACT and

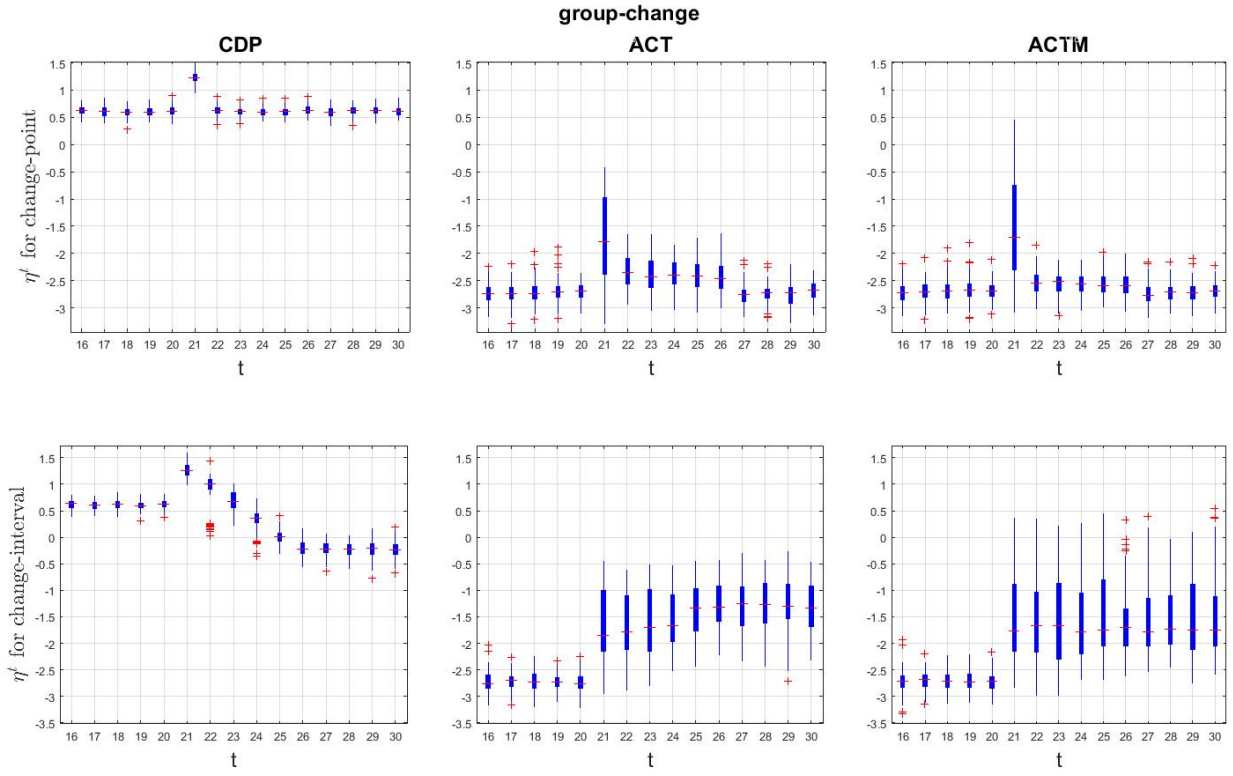


FIGURE 3.2: **Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on group-change for $w = 5$ change point(top) and change-interval (bottom).** CDP shows a clear detection at $t = 21$ for both change point and change-interval. Although ACT and ACTM methods also show an increase at $t = 21$, the intervals still lie below zero.

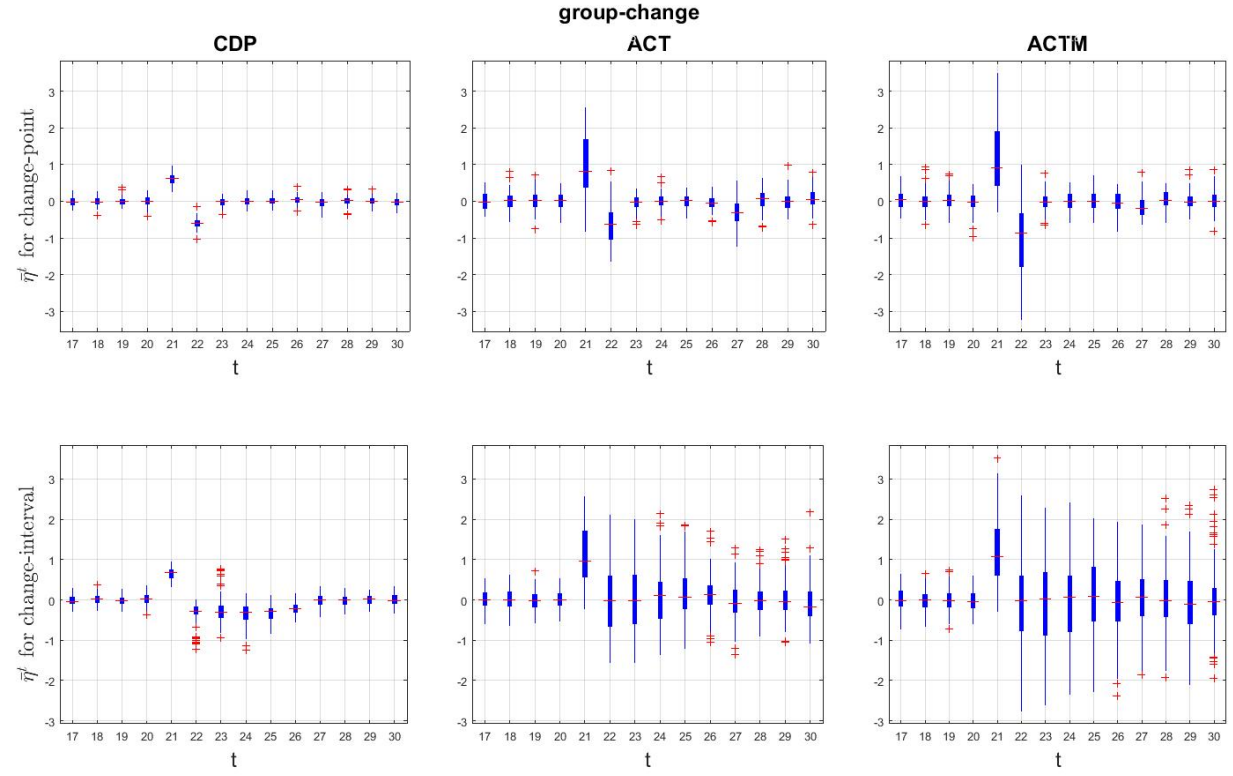


FIGURE 3.3: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on group-change for $w = 5$ change point(top) and change-interval (bottom).**

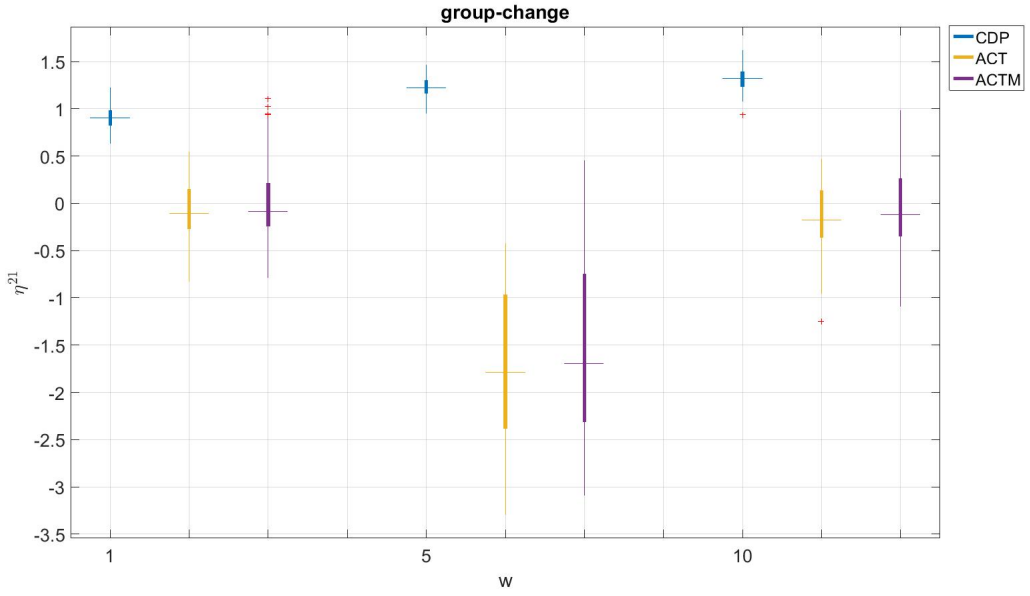


FIGURE 3.4: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for group-change for $w = 1, 5, 10$ at change-point**. For all w, $\boldsymbol{\eta}^{21}{}_{CDP}$ is positive and increases with w. For all w, a majority of elements of $\boldsymbol{\eta}^{21}{}_{ACT}$ and $\boldsymbol{\eta}^{21}{}_{ACTM}$ are negative.

ACTM. For fragment, $\bar{\boldsymbol{\eta}}^{21}$ is highly negative for both ACT and ACTM methods. Thus, although we observed in Figure 3.11 that ACT shows good performance in terms of $\boldsymbol{\eta}^{21}$, Figure 3.12 shows that the change scores have decreased at $t = 21$. Thus, ACT shows failure in detecting the fragment scenario.

In Table 3.3, we perform the sign test to assess the statistical significance of the observed results. We compare $\boldsymbol{\eta}^{21}$ calculated for group-change, split, merge, form, hetero-to-homo, homo-to-hetero, simple-to-complex, and complex-to-simple at $w = 5$ for change-point. We do not perform the sign test for fragment scenario as we already observed a decrease in $\bar{\boldsymbol{\eta}}^{21}$ for ACT and ACTM compared to previous time instants in Figure 3.12 (this clearly shows how CDP outperforms these two methods). The leftmost column in Table 3.3 gives the alternative hypothesis tested. Subsequently in Table 3.4, we show the proportion of values in $\boldsymbol{\eta}^{21}$, that correspond to the hypothesis tested in Table 3.3. CDP outperforms ACT and ACTM for all change scenarios except form. ACTM outperforms ACT for group-change, form, and homo-to-hetero. For the other scenarios tested, there is no difference in $\boldsymbol{\eta}^{21}$ for ACT and ACTM. However, when we consider the proportions in Table 3.4, the majority of the entries in $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$ are greater than $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ for all change scenarios except hetero-to-homo. From the overall simulation results we see that, while ACTM is slightly better than ACT, CDP is the best of the three.

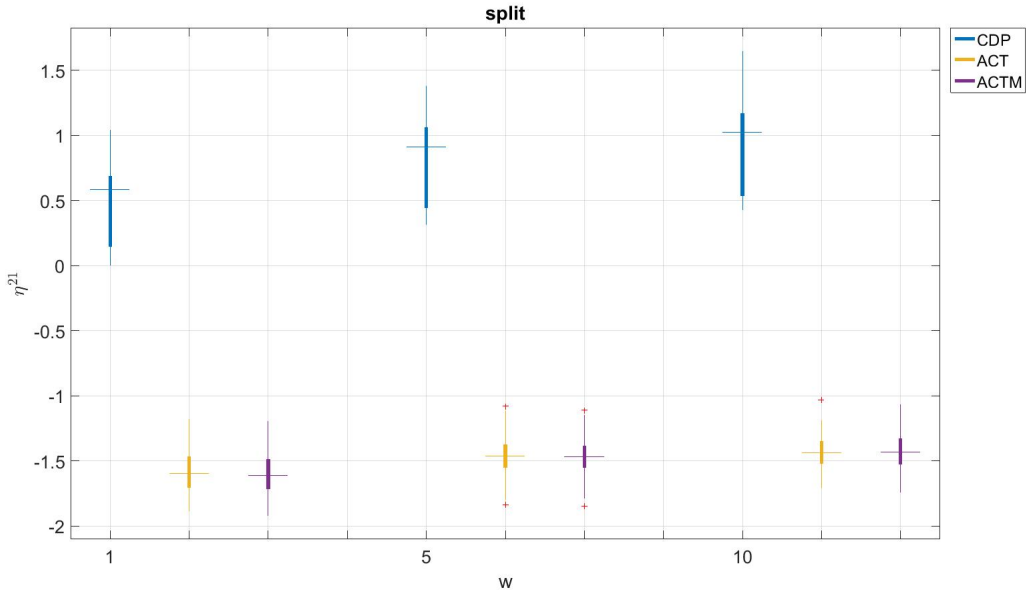


FIGURE 3.5: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for split for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$ is positive and increases with w. For all w, ${\boldsymbol{\eta}^{21}}_{ACT}$ and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are negative.

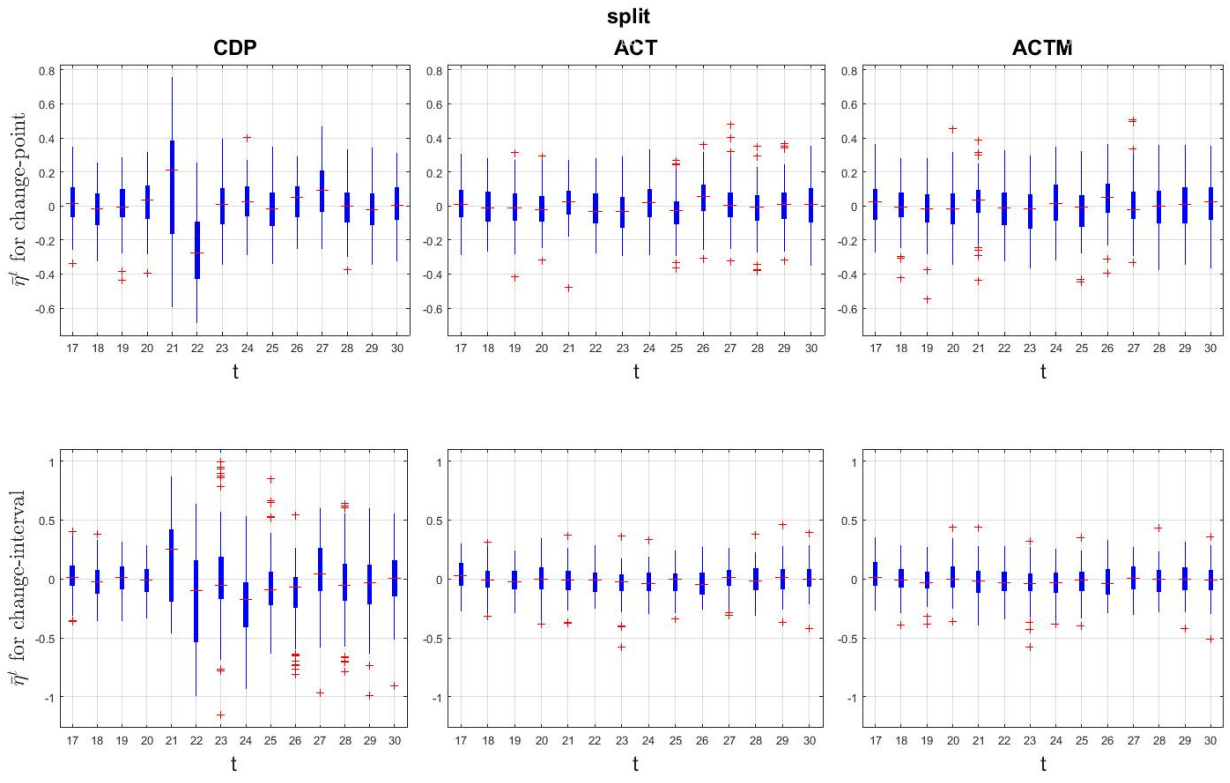


FIGURE 3.6: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on split for $w = 5$ change point(top) and change-interval (bottom)**. Although $\bar{\boldsymbol{\eta}}^t_{CDP}$ shows an increase at $t = 21$ for both change point and change-interval, $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ shows high variability further extending below zero. ACT and ACTM do not show an increase at $t = 21$.

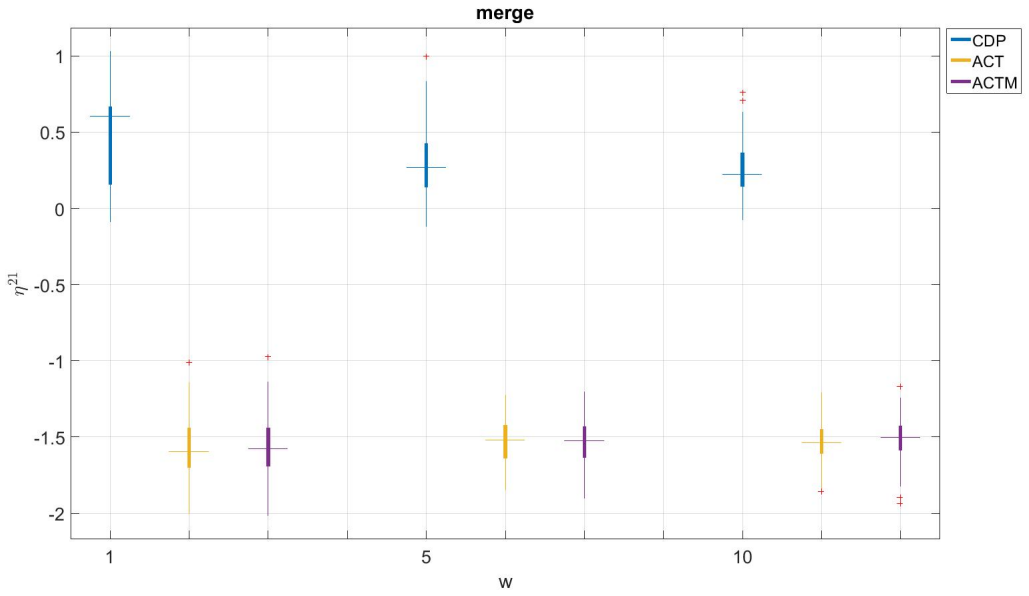


FIGURE 3.7: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for merge for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$ is positive and decreases with w. For all w, ${\boldsymbol{\eta}^{21}}_{ACT}$ and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are negative.

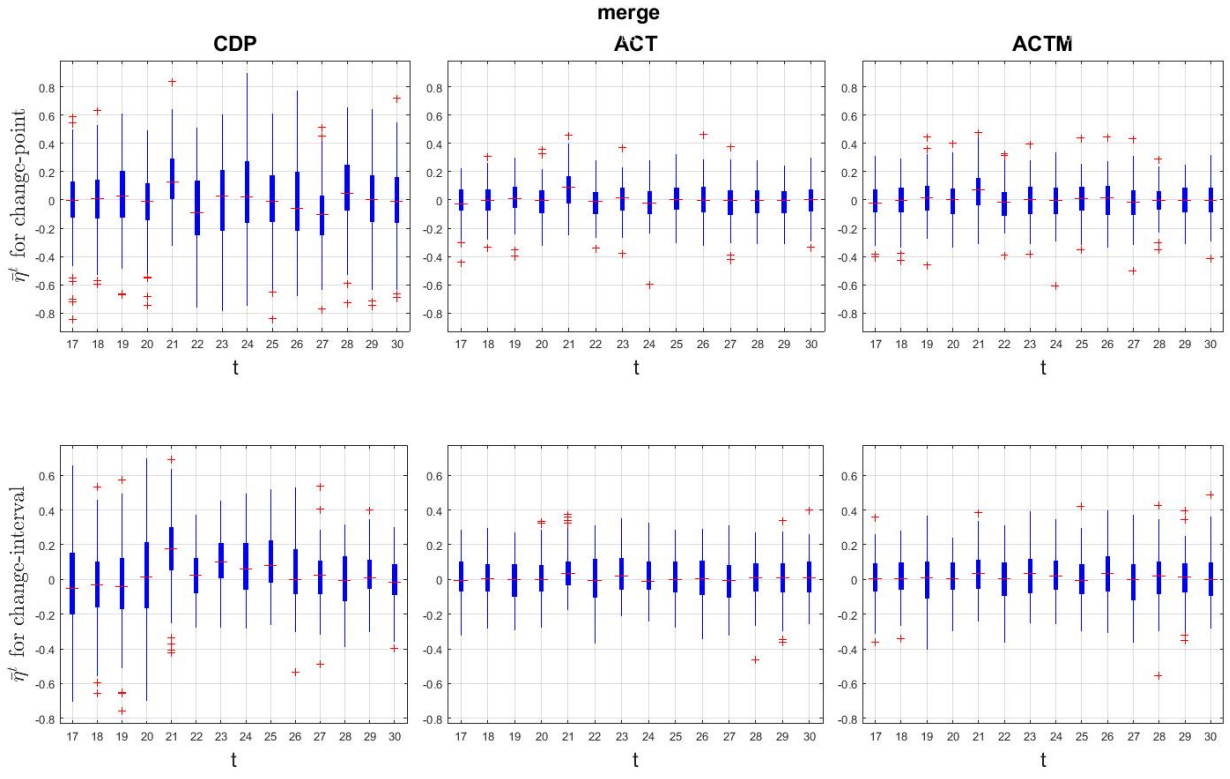


FIGURE 3.8: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on merge for $w = 5$ change point(top) and change-interval (bottom)**. Although $\bar{\boldsymbol{\eta}}^t_{CDP}$ shows an increase at $t = 21$ for both change point and change-interval, $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ shows high variance, further extending below zero. ACT and ACTM do not show a clear increase at $t = 21$ for both change-point and change-interval.

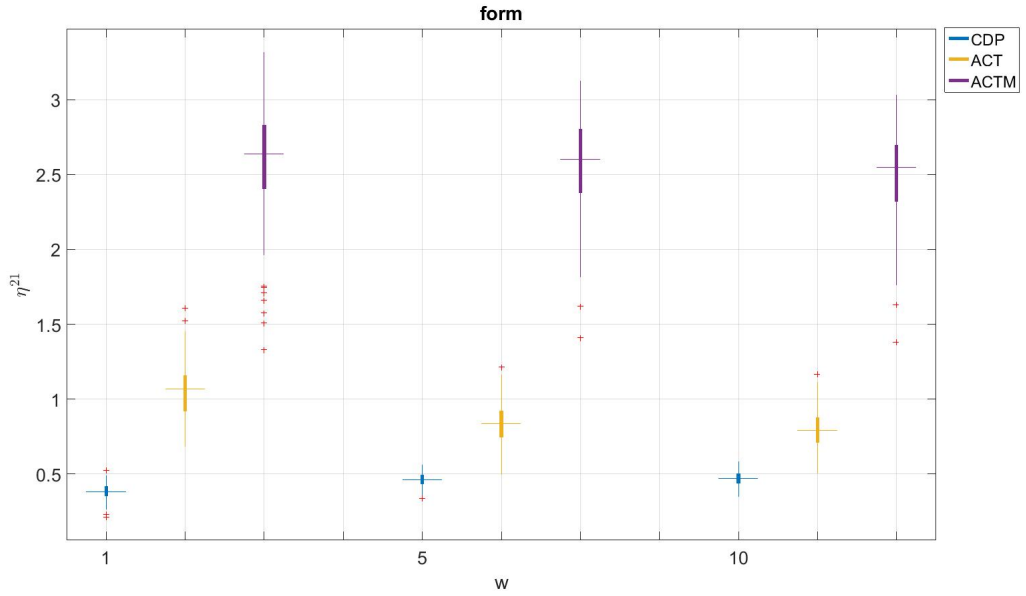


FIGURE 3.9: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for form for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$, ${\boldsymbol{\eta}^{21}}_{ACT}$, and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are positive and increase with w. However, ${\boldsymbol{\eta}^{21}}_{ACTM}$ is wider and has more outliers.

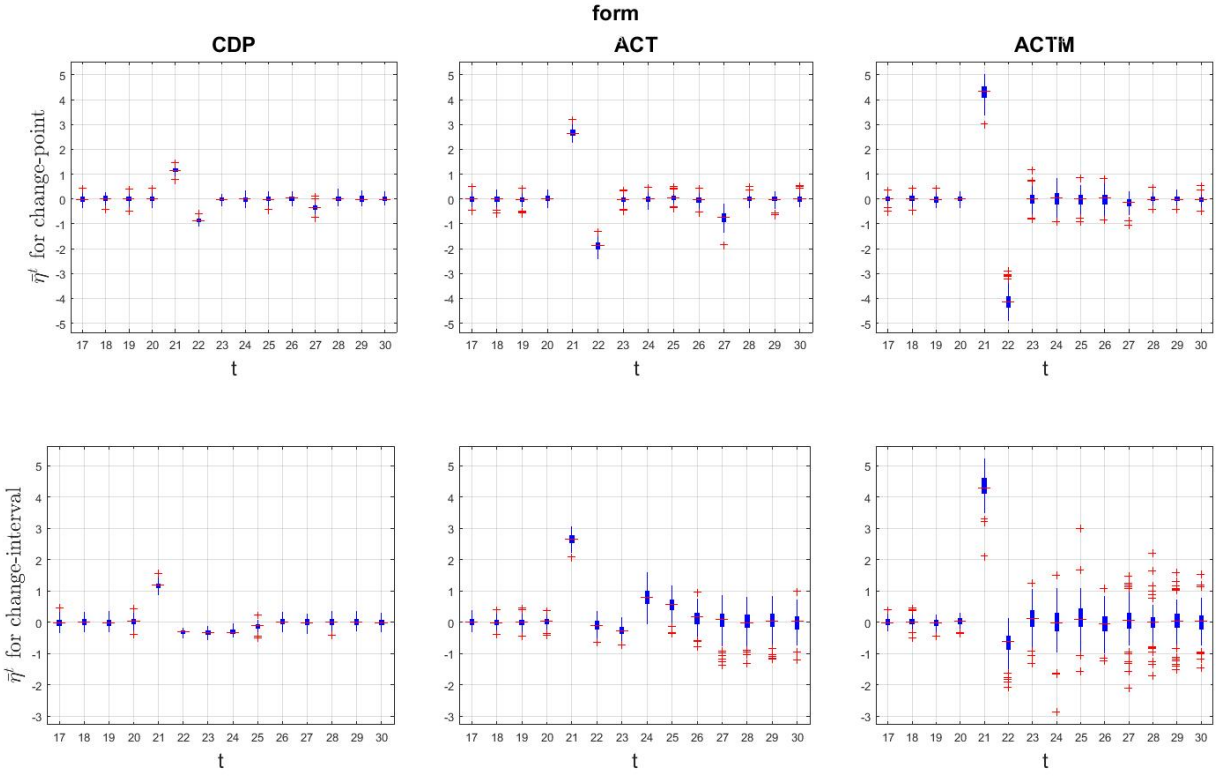


FIGURE 3.10: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on form for $w = 5$ change point(top) and change-interval (bottom)**. All $\bar{\boldsymbol{\eta}}^t_{CDP}$, $\bar{\boldsymbol{\eta}}^t_{ACT}$ and $\bar{\boldsymbol{\eta}}^t_{ACTM}$ show a clear increase at $t = 21$, while $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$ shows the highest increase.

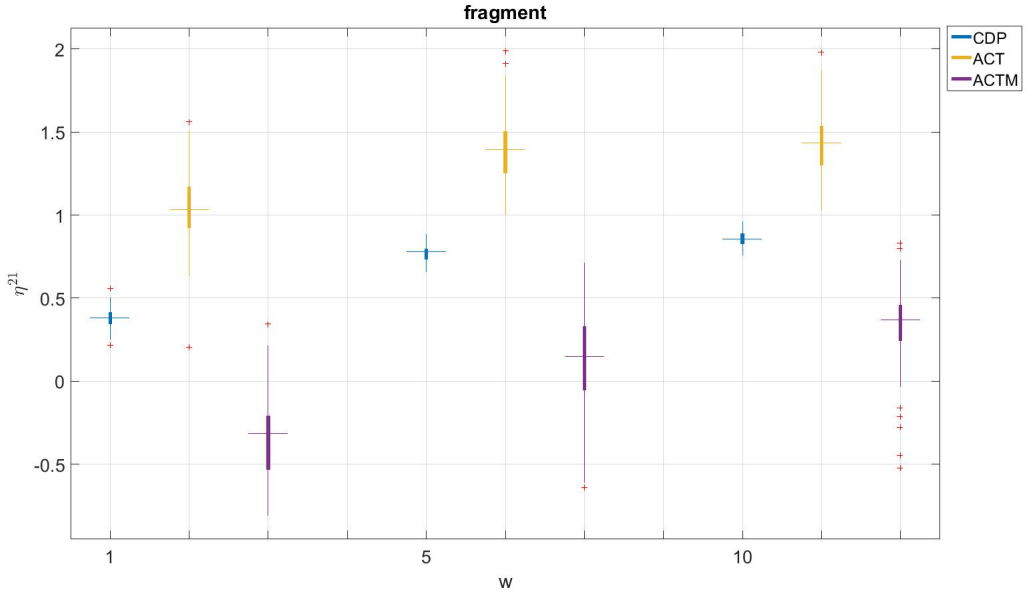


FIGURE 3.11: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for fragment for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$ and ${\boldsymbol{\eta}^{21}}_{ACT}$ are positive and increase with w. ${\boldsymbol{\eta}^{21}}_{ACTM}$ is mostly negative at $w = 1$, but increases with w. However, ${\boldsymbol{\eta}^{21}}_{ACT}$ and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are wider and show more outliers.

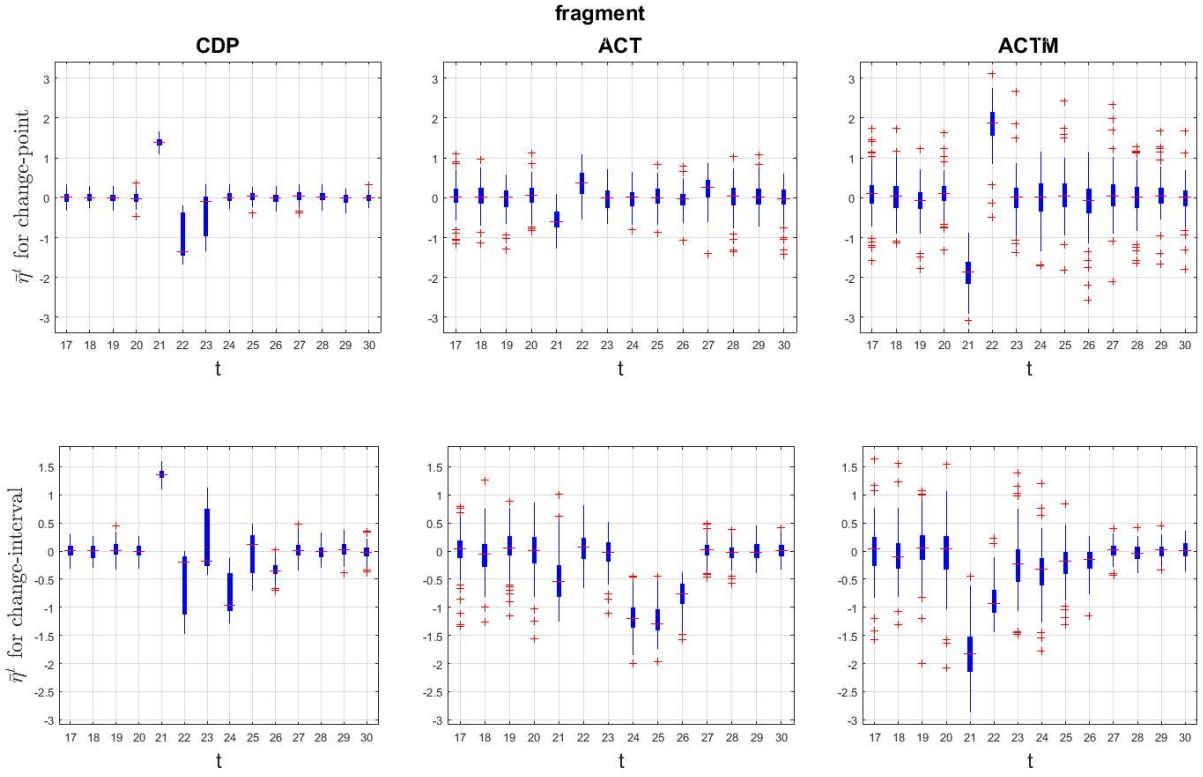


FIGURE 3.12: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on fragment for $w = 5$ change point(top) and change-interval (bottom)**. $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ shows a clear detection at $t = 21$ for both change point and change-interval. Both $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$ are negative. Furthermore $\bar{\boldsymbol{\eta}}^{t}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{t}_{ACTM}$ contain a large number of outliers.

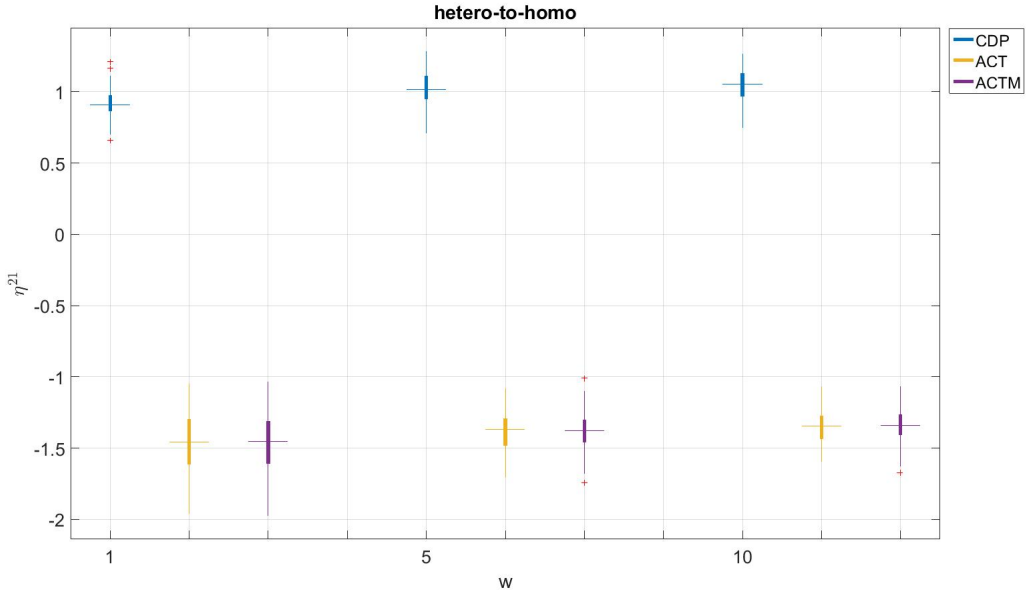


FIGURE 3.13: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for hetero-to-homo for $w = 1, 5, 10$ at change-point**. For all w, $\boldsymbol{\eta}^{21}{}_{CDP}$ are positive and slightly increase with w. The $\boldsymbol{\eta}^{21}{}_{ACT}$ and $\boldsymbol{\eta}^{21}{}_{ACTM}$ are negative for all w.

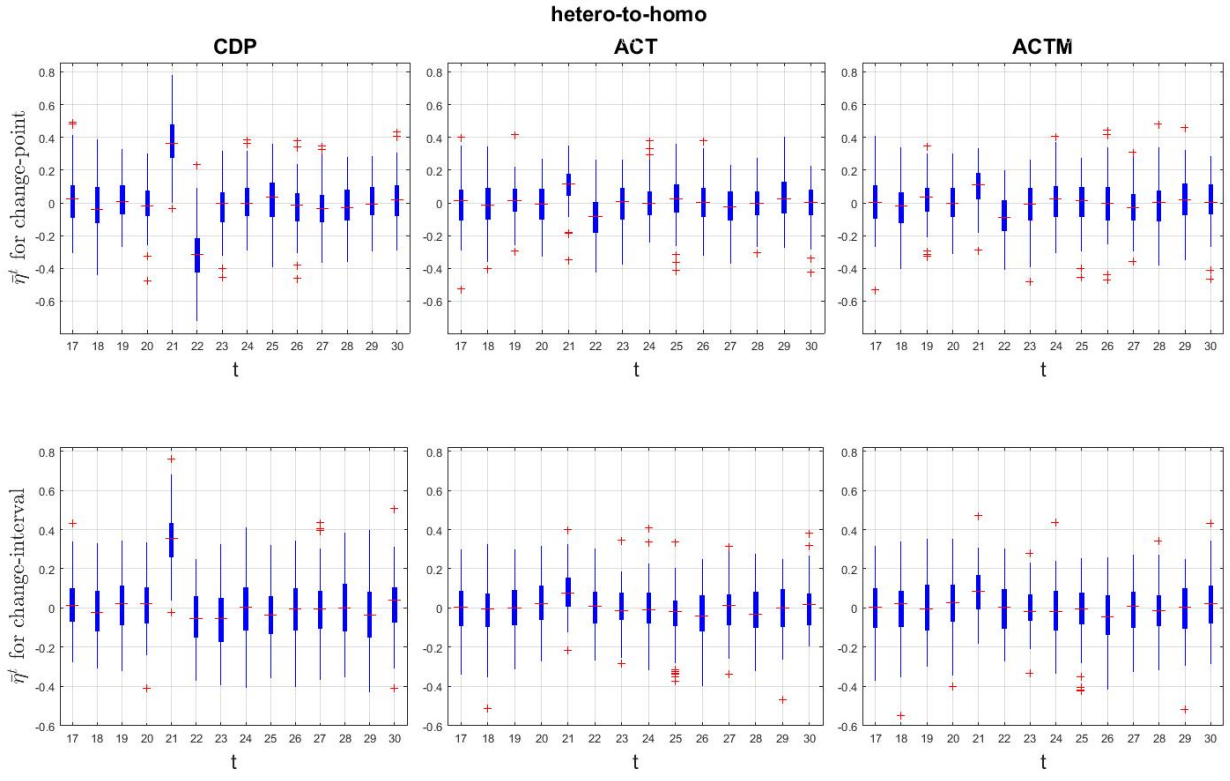


FIGURE 3.14: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on hetero-to-homo for $w = 5$ change point(top) and change-interval (bottom)**. $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ shows a clear detection at $t = 21$ for both change point and change-interval. Both $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$ show a slight increase, but still some values lie below zero.

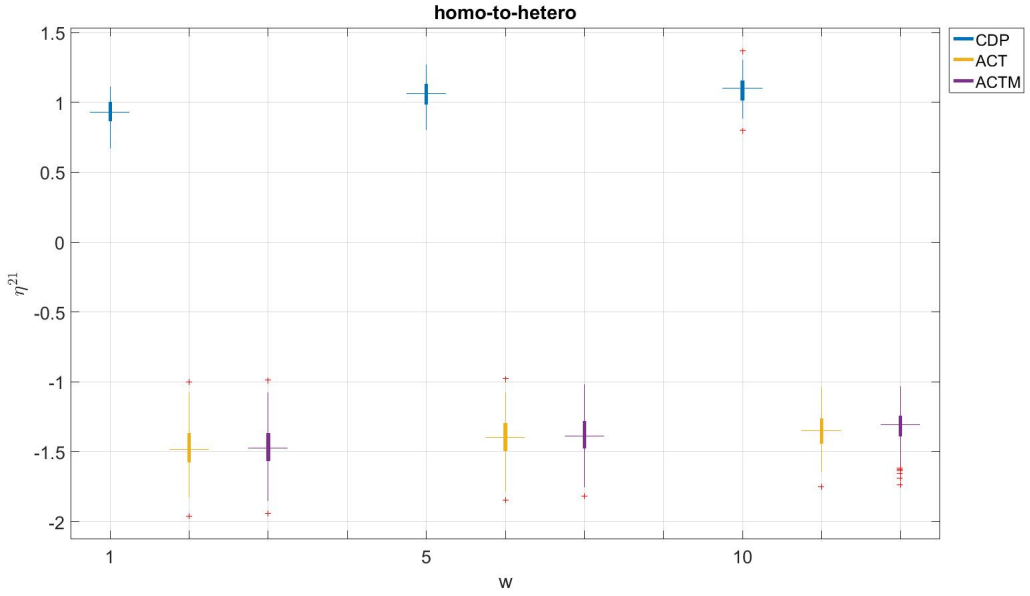


FIGURE 3.15: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for homo-to-hetero for $w = 1, 5, 10$ at change-point**. For all w, $\boldsymbol{\eta}^{21}{}_{CDP}$ are positive and slightly increase with w. The $\boldsymbol{\eta}^{21}{}_{ACT}$ and $\boldsymbol{\eta}^{21}{}_{ACTM}$ are negative for all w.

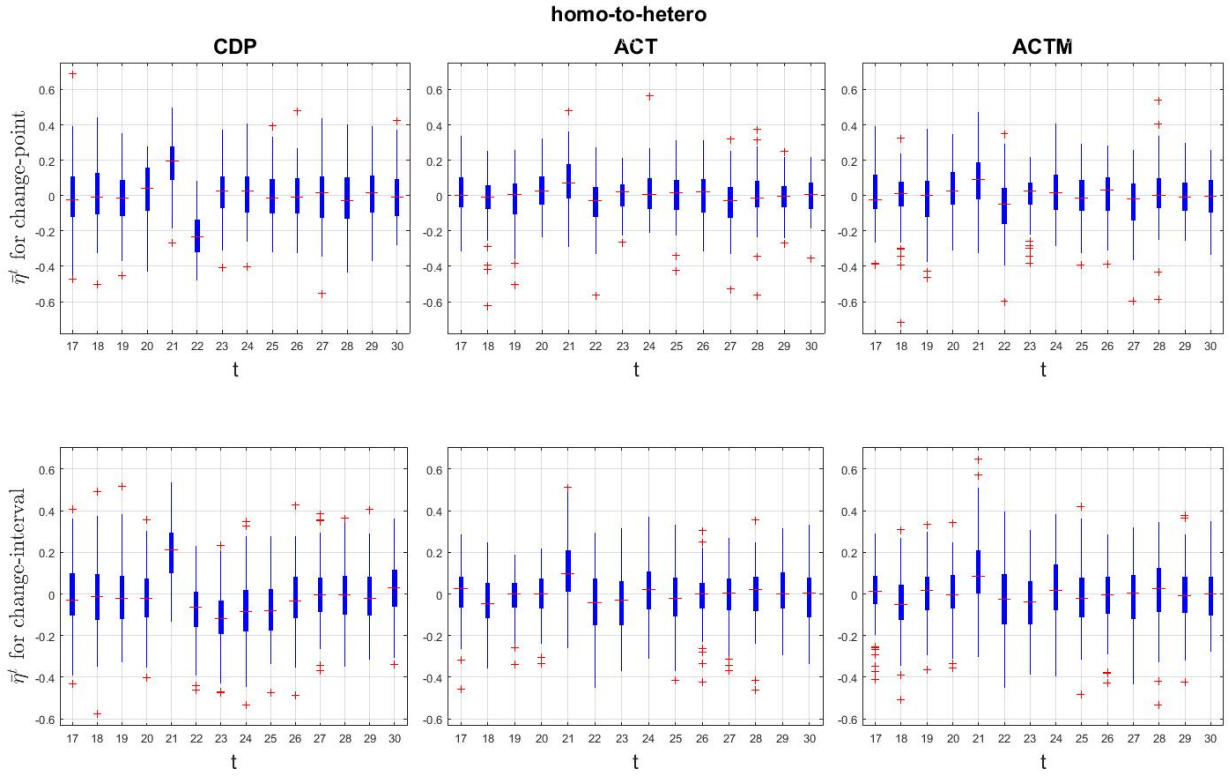


FIGURE 3.16: **Observing $\bar{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on homo-to-hetero for $w = 5$ change point(top) and change-interval (bottom)**. $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ shows a clear detection at $t = 21$ for both change point and change-interval, but $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ slightly extends below zero. Both $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$ also show a slight increase, with the intervals extending below zero.

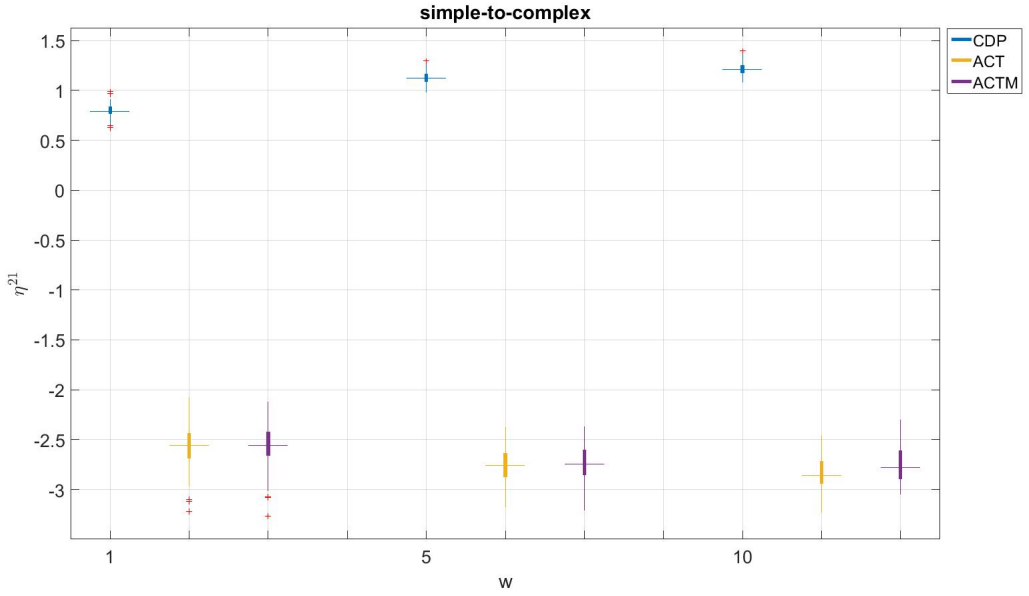


FIGURE 3.17: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for simple-to-complex for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$ are positive and increase with w. The ${\boldsymbol{\eta}^{21}}_{ACT}$ and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are negative for all w.

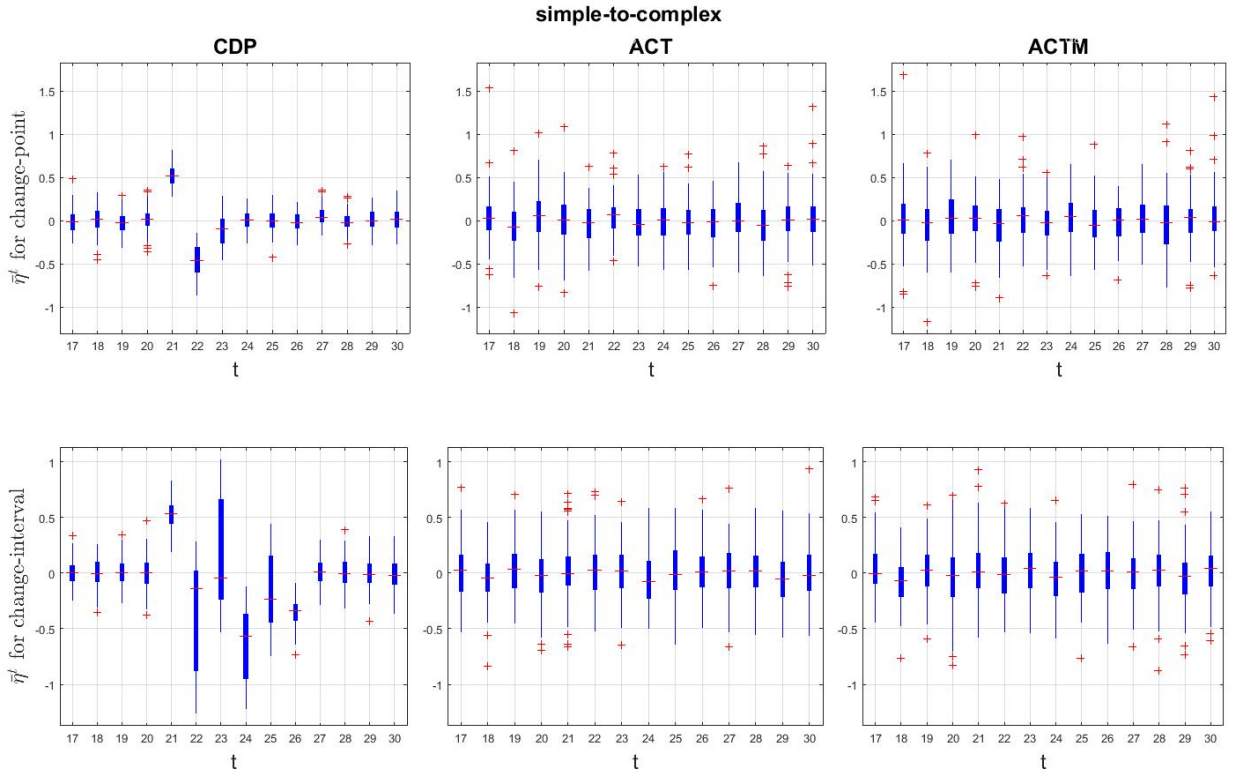


FIGURE 3.18: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on simple-to-complex for $w = 5$ change point(top) and change-interval (bottom)**. We observe a clear detection at $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ for change-point. For change-interval we observe $\bar{\boldsymbol{\eta}}^{t}_{CDP}$ for $t = 22, \ldots, 25$ to have high variance. A detection is not observed on both $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$.

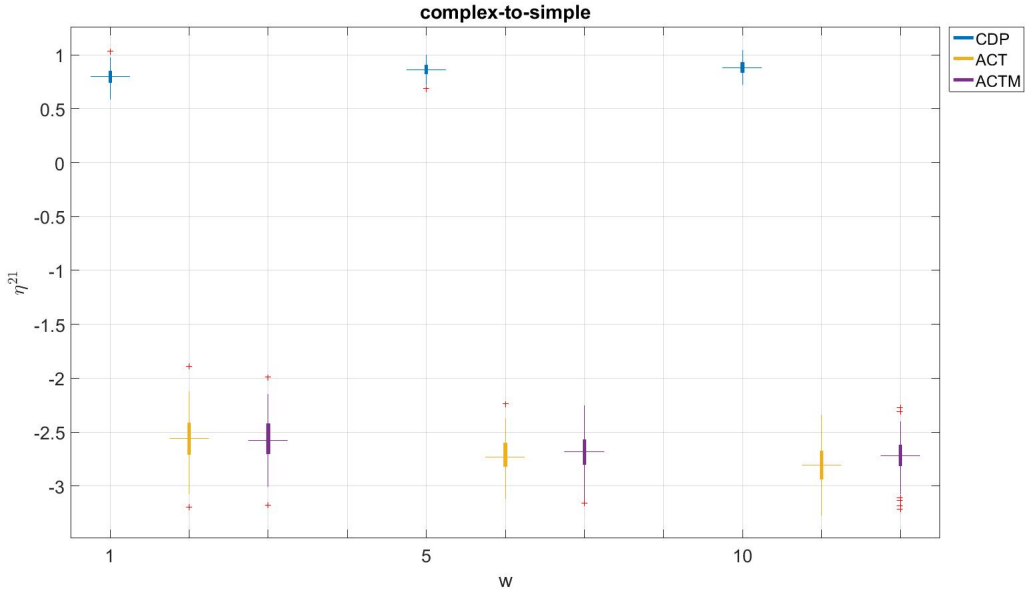


FIGURE 3.19: **Plot of $\boldsymbol{\eta}^{21}$ on CDP, ACT, and ACTM for complex-to-simple for $w = 1, 5, 10$ at change-point**. For all w, ${\boldsymbol{\eta}^{21}}_{CDP}$ are positive and slightly increase with w. The ${\boldsymbol{\eta}^{21}}_{ACT}$ and ${\boldsymbol{\eta}^{21}}_{ACTM}$ are negative for all w.

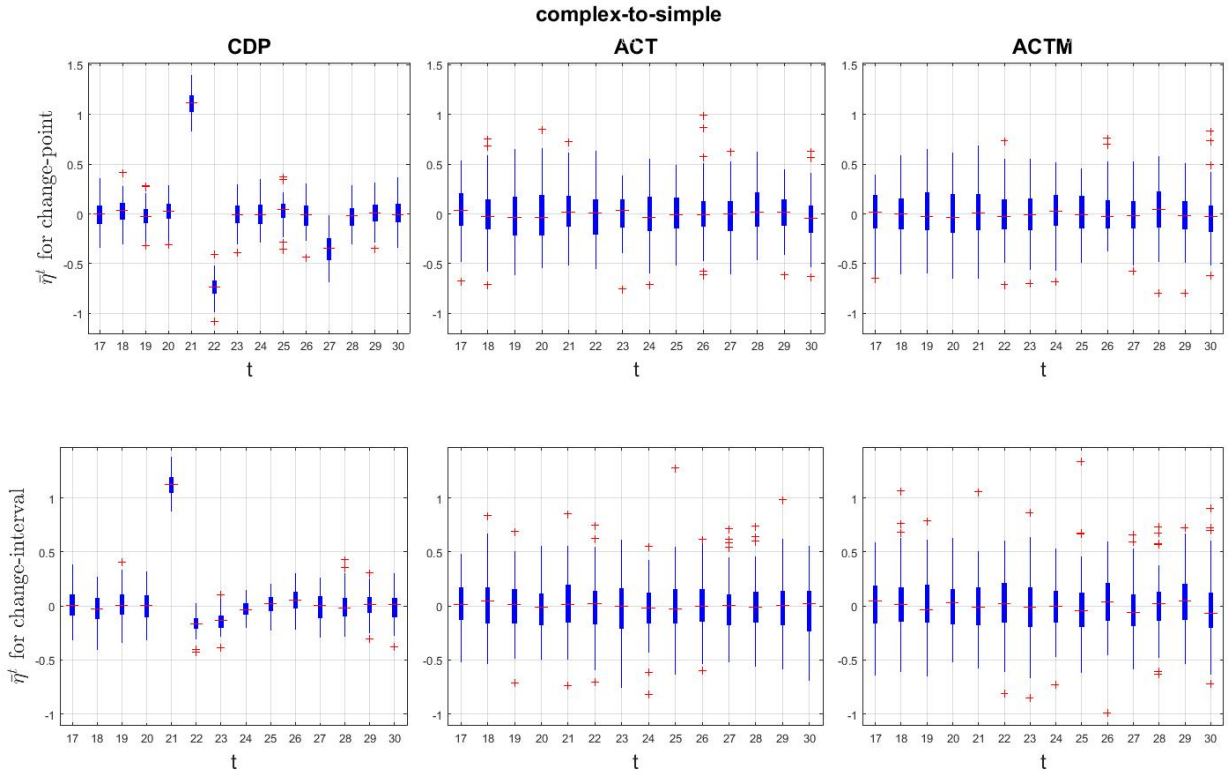


FIGURE 3.20: **Observing $\bar{\boldsymbol{\eta}}^t$ on CDP (left), ACT (middle), and ACTM (right) over time on complex-to-simple for $w = 5$ change point(top) and change-interval (bottom)**. We observe a clear detection at $\bar{\boldsymbol{\eta}}^{21}_{CDP}$ for both change-point and change-interval. A detection is not observed on both $\bar{\boldsymbol{\eta}}^{21}_{ACT}$ and $\bar{\boldsymbol{\eta}}^{21}_{ACTM}$.

TABLE 3.3: Sign test results for comparing $\boldsymbol{\eta}^{21}$ calculated using CDP, ACT, and ACTM at $w = 5$ for change-point.

Alternative Hypothesis	group-change	split	merge	form	homo-to-hetero	hetero-to-homo	simple-to-complex	complex-to-simple
$\boldsymbol{\eta}^{21}_{CDP} > \boldsymbol{\eta}^{21}_{ACT}$	2.08e-23	2.08e-23	2.08e-23	1.0000	2.08e-23	2.08e-23	2.08e-23	2.08e-23
$\boldsymbol{\eta}^{21}_{CDP} < \boldsymbol{\eta}^{21}_{ACT}$	1.0000	1.0000	1.0000	2.08e-23	1.0000	1.0000	1.0000	1.0000
$\boldsymbol{\eta}^{21}_{CDP} \neq \boldsymbol{\eta}^{21}_{ACT}$	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23
$\boldsymbol{\eta}^{21}_{CDP} > \boldsymbol{\eta}^{21}_{ACTM}$	2.08e-23	2.08e-23	2.08e-23	1.0000	2.08e-23	2.08e-23	2.08e-23	2.08e-23
$\boldsymbol{\eta}^{21}_{CDP} < \boldsymbol{\eta}^{21}_{ACTM}$	1.0000	1.0000	1.0000	2.08e-23	1.0000	1.0000	1.0000	1.0000
$\boldsymbol{\eta}^{21}_{CDP} \neq \boldsymbol{\eta}^{21}_{ACTM}$	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23	4.16e-23
$\boldsymbol{\eta}^{21}_{ACTM} > \boldsymbol{\eta}^{21}_{ACT}$	4.02e-11	0.3085	0.3821	2.08e-23	0.00347	0.6915	0.2420	0.0968
$\boldsymbol{\eta}^{21}_{ACTM} < \boldsymbol{\eta}^{21}_{ACT}$	1.0000	0.7580	0.6915	1.0000	0.9981	0.3821	0.8159	0.9332
$\boldsymbol{\eta}^{21}_{ACTM} \neq \boldsymbol{\eta}^{21}_{ACT}$	8.03e-11	0.6171	0.7642	4.16e-23	0.0069	0.7642	0.4840	0.1936

TABLE 3.4: Proportions for comparing $\boldsymbol{\eta}^{21}$ calculated using CDP, ACT, and ACTM at $w = 5$ for change-point

Categories	group-change	split	merge	form	homo-to-hetero	hetero-to-homo	simple-to-complex	complex-to-simple
$\boldsymbol{\eta}^{21}_{CDP} > \boldsymbol{\eta}^{21}_{ACT}$	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
$\boldsymbol{\eta}^{21}_{CDP} < \boldsymbol{\eta}^{21}_{ACT}$	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
$\boldsymbol{\eta}^{21}_{CDP} = \boldsymbol{\eta}^{21}_{ACT}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\boldsymbol{\eta}^{21}_{CDP} > \boldsymbol{\eta}^{21}_{ACTM}$	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
$\boldsymbol{\eta}^{21}_{CDP} < \boldsymbol{\eta}^{21}_{ACTM}$	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
$\boldsymbol{\eta}^{21}_{CDP} = \boldsymbol{\eta}^{21}_{ACTM}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\boldsymbol{\eta}^{21}_{ACTM} > \boldsymbol{\eta}^{21}_{ACT}$	0.83	0.53	0.52	1.00	0.64	0.48	0.54	0.57
$\boldsymbol{\eta}^{21}_{ACTM} < \boldsymbol{\eta}^{21}_{ACT}$	0.17	0.47	0.48	0.00	0.36	0.52	0.46	0.43
$\boldsymbol{\eta}^{21}_{ACTM} = \boldsymbol{\eta}^{21}_{ACT}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Each change scenario discussed in Section 3.3, consists of a change in the behaviour of a subset of vertices in the DCSBM graph: (i) split, merge, homo-to-hetero, and hetero-to-homo involve the set of vertices, $\{v_1, v_2, \ldots, v_{300}\}$, in the most sparsely connected block in the graph, (ii) group-change, simple-to-complex, and complex-to-simple involve the set of vertices, $\{v_1, v_2, \ldots, v_{600}\}$, that can be considered to be moderately connected in the graph, and (iii) form and fragment involve the set of vertices, $\{v_{601}, v_{602}, \ldots, v_{900}\}$, from the most dense block in the graph. From the results of the experiments conducted, we observe how CDP detects changes involving each of these subsets of vertices. ACT and ACTM could only detect the form scenario that involves a change in the vertices from the most dense block in the graph. However still, ACT and ACTM failed to detect fragment, which is the reverse of form.

The parameter, λ, in our DCSBM (Equation 3.15) enables us to control the level of noise included in a generated graph. When $\lambda = 0$, the graph will have a fully random structure with no blocks. When $\lambda = 1$ the graph will have clear-cut blocks, making it easy for the embeddings to identify the structure. In Figure 3.21, we observe $\boldsymbol{\eta}^{21}_{CDP}$ by varying λ from zero to one for all change-point scenarios with $w = 5$. We observe that the $\boldsymbol{\eta}^{21}_{CDP}$'s are generally positive for $\lambda > 0.6$ and follow an increasing or non-decreasing pattern. Thus, CDP generally performs well for $\lambda = 0.6$ or higher.

Next, we conduct experiments to evaluate the scalability of CDP and the other two baseline methods. We select one change scenario, and generate a sequence of six graphs. At the sixth time instant, we calculate vertex change scores using a window of size five. We repeat this over 100 simulation runs. We calculate the average CPU time taken to embed a single graph, and the average CPU time taken to calculate profile behaviour and change scores at a given time instant. Following the same procedure, we conduct experiments on graphs with several sizes. All experiments are implemented on a Windows server Intel Xeon with two 3.3GHz processors of 128 GB RAM. Our results are given in Table 3.5. Figures 3.22 and 3.23 plot the average computational time taken by each method for the embedding step and change score calculation step, respectively for different graph sizes.

The most computationally efficient methods for embedding a graph are the ACT and ACTM methods. These methods both perform SVD of the weighted adjacency matrix to extract a single singular vector. However, as shown in our previous results, keeping only one singular vector does not provide a good representation of all vertices in the network. CDP performs SVD to extract d singular vectors from the representation matrix to obtain an optimal representation, hence taking more computational time. When comparing change score calculation times (Figure 3.23), CDP is clearly more

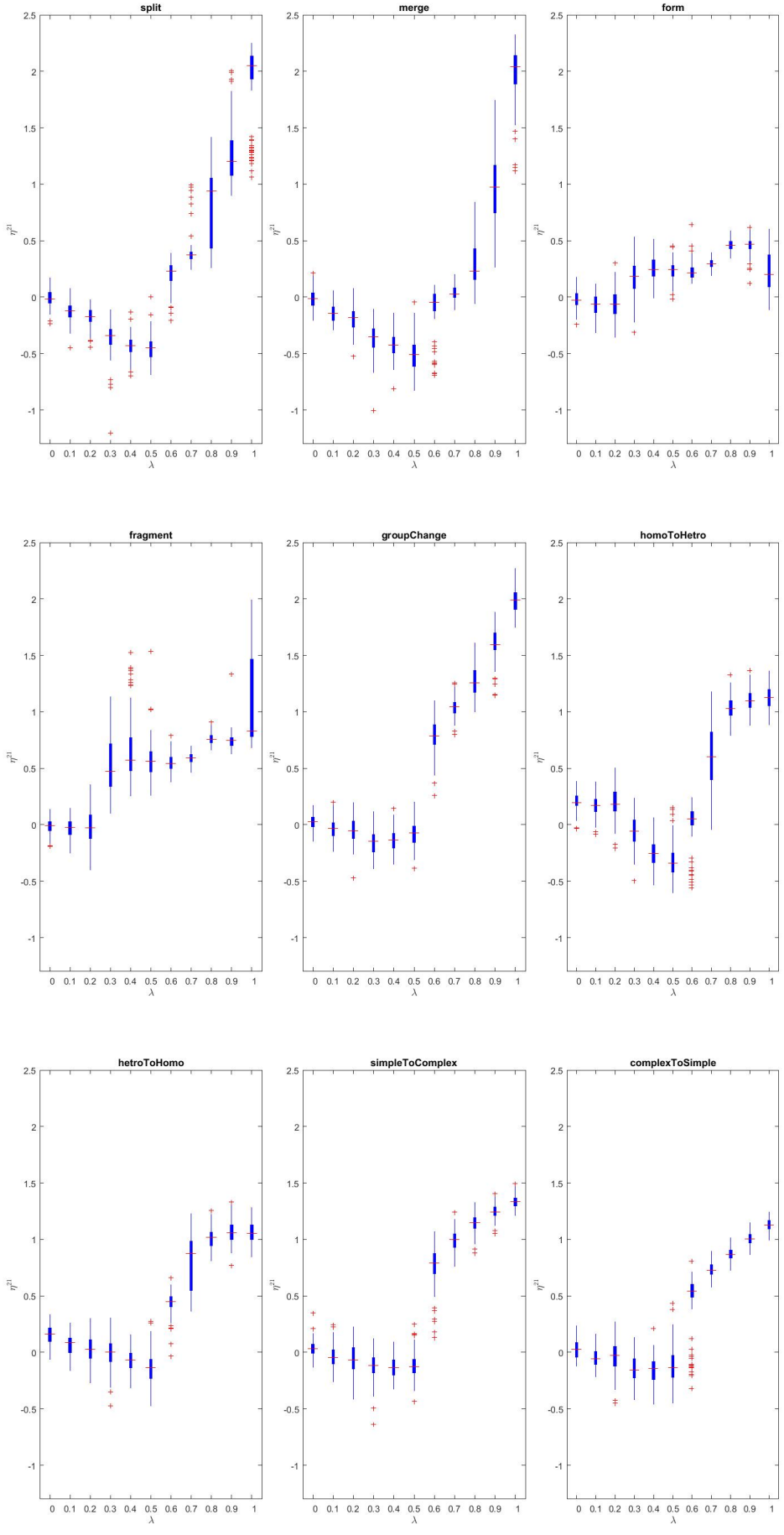


FIGURE 3.21: **Comparison of $\boldsymbol{\eta}^{21}_{CDP}$ by varying the noise level, λ, of generated graphs.** For all change scenarios, $\boldsymbol{\eta}^{21}_{CDP}$ is generally positive for $\lambda \geq 0.6$.

TABLE 3.5: Average CPU time (seconds) taken by each method

Task	n	CDP	ACT	ACTM
Spectral embedding	300	0.2425	0.0474	0.0475
	900	1.7584	0.4554	0.4832
	1500	4.6635	1.4223	1.5394
	2100	11.8364	4.5364	3.7637
	2700	20.2857	9.8368	7.3115
	3300	51.3395	22.0778	18.0710
Profile behaviour and change score calculation	300	0.0063	0.0032	0.0026
	900	0.0142	0.0156	0.0144
	1500	0.0340	0.0546	0.0570
	2100	0.0725	0.1095	0.1140
	2700	0.1016	0.1757	0.1817
	3300	0.1739	0.2394	0.2382

efficient than ACT or ACTM for graph sizes greater than 900. However, we observe that the change score times are negligible compared to the respective embedding times.

From the overall simulation results, it can be concluded that CDP method outperforms the other two baseline methods, and is the most reliable method in detecting the different types of change scenarios considered.

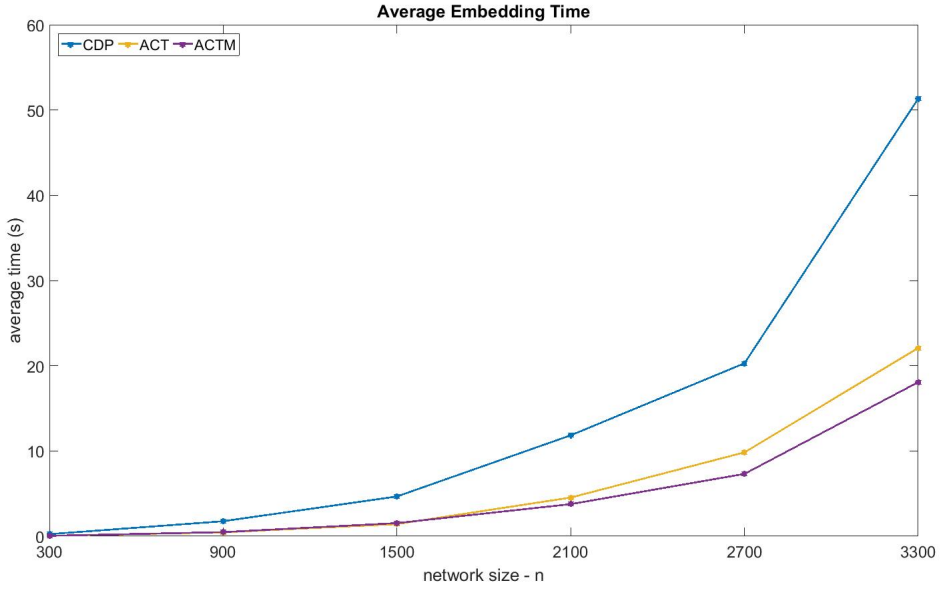


FIGURE 3.22: Comparison of average CPU time taken to embed a graph using different methods.

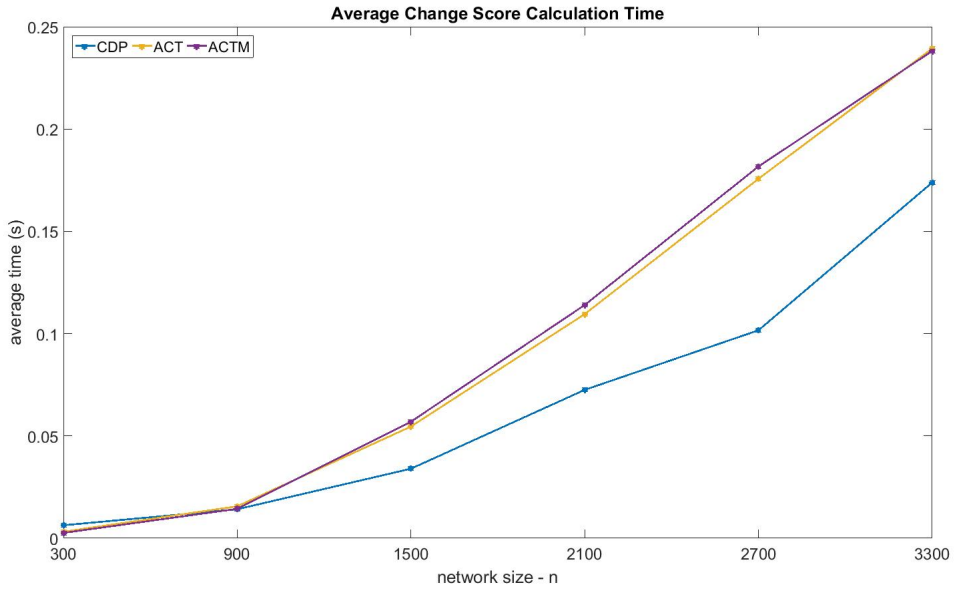


FIGURE 3.23: Comparison of average CPU time taken to calculate change scores using different methods.

3.4 Case Study: The Enron E-mail Network

The Enron email dataset (Klimt and Yang, 2004) provides information regarding emails exchanged in the Enron company. Enron was declared bankrupt in 2002 due to accountancy fraud activities by the top level executives. These email data were made publicly available[3] by the Federal Energy Regulatory during their investigations of the famous Enron fraud. The Enron dataset is used in various publications for community detection and anomaly detection (Peel and Clauset, 2015; Priebe et al., 2005; Rossi et al., 2013). In this thesis, we use the cleaned and processed version of the dataset created by Tang et al. (2008). Based on the sent and received email addresses in the original Enron corpus, Tang et al. extract a total of 2359 user email addresses, and construct a time sequence of email-sender networks and a time sequence of email-receivers networks for each month from December 1999 to March 2002. Based on these two dynamic networks, we construct a time sequence of 28 undirected graphs (one for each month), where the vertices denote user email addresses, and the edge weights denote the number of emails communicated (either sent or received) by the corresponding pair of users. Each graph is then represented by an $n \times n$ symmetric weighted adjacency matrix, where $n = 2359$.

We applied CDP, ACT, and ACTM on this data with a window of length 1 (we used $w = 1$ as the time instants correspond to months). Each of the methods, CDP, ACT, and ACTM returns an $n \times 1$ vector of vertex change scores for each time instant. Our

[3] http://www.cs.cmu.edu/~enron/

goal is to detect vertices which have changed most during a given time instant. To achieve this goal, we employ the following simple procedure: Let $\mathbf{z}^t$ be the vector of vertex change scores obtained for a given method. Each z_i^t is converted to a z-score, $\hat{z}_i^t = \frac{(z_i^t - \bar{z}^t)}{\sigma_{\mathbf{z}^t}}$, where $\bar{z}^t = \frac{1}{n}\sum_{i=1}^{n} z_i^t$, and $\sigma_{\mathbf{z}^t}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(z_i^t - \bar{z}^t)^2$. We threshold each z-score distribution, $\hat{\mathbf{z}}^t$, at 5 to detect the vertices which changed the most at that time instant (we investigate and find that the threshold 5 for this dataset ensures the percentage of vertices detected at each time instant is less than two percent for all three methods). The Enron time-line[4] contains a detailed description of the key players[5], and events that took place during the rise and fall of the Enron company. Based on the assumption that the email communication patterns within the company were affected by the events associated with the scandal, we evaluate the performance of our change detection method.

We find Timothy Beldon (chief trader of Enron's West Coast power desk and convicted of wire fraud) to be one of the entities which changed the most, using CDP for the time instants corresponding to September-2000 and October-2000. According to the Enron time-line, this is the time when an attorney from Enron travelled to Portland to discuss Timothy Beldon's strategies of boosting energy prices. Figure 3.24 shows the rate of change in the number of emails sent and received by Beldon throughout the whole time period considered. First, in September-2000, we observe a noticeable drop in the number of emails communicated, which then suddenly increases in October-2000. Figures 3.25, 3.26, and 3.27 further show the subgraphs consisting of the vertices corresponding to Timothy Beldon and his connections for time instants, August-2000, September-2000 and October-2000 respectively. It is observed that Beldon, who communicated with many employees in various job roles in August-2000, limited his communications mostly to the top level executives and CEO's during September-2000. He then starts communicating with many employees in different job roles in October-2000. CDP successfully detects this change in degree as well as community membership by giving high change scores to Timothy Beldon during the respective time instants.

Enron announced that current CEO Kenneth Lay was to be replaced by Jeffry Skilling in December-2000. CDP successfully detects Rosalie Fleming, who was the assistant of Kenneth Lay, as one of the top changed entities for this time instant. The subgraph in Figure 3.28 show Fleming's connections in November-2000 which are mostly with the employees in the company. Figure 3.29 shows how she starts communicating with people with different job roles such as CEO's, directors, etc. in December-2000. We

[4] http://www.agsm.edu.au/bobm/teaching/BE/Enron/timeline.html

[5] http://www.finanznachrichten.de/nachrichten-2006-08/6810532-key-players-in-enron-scandal-020.htm

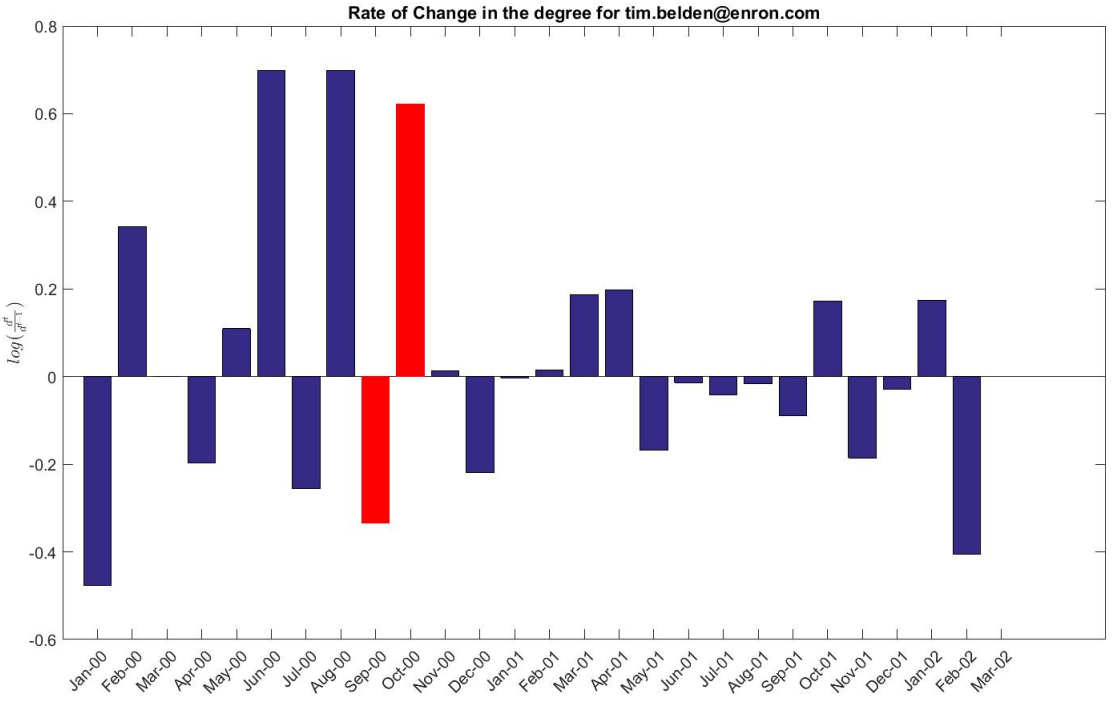


FIGURE 3.24: Rate of change in the number of emails sent and received by Timothy Beldon between consecutive months

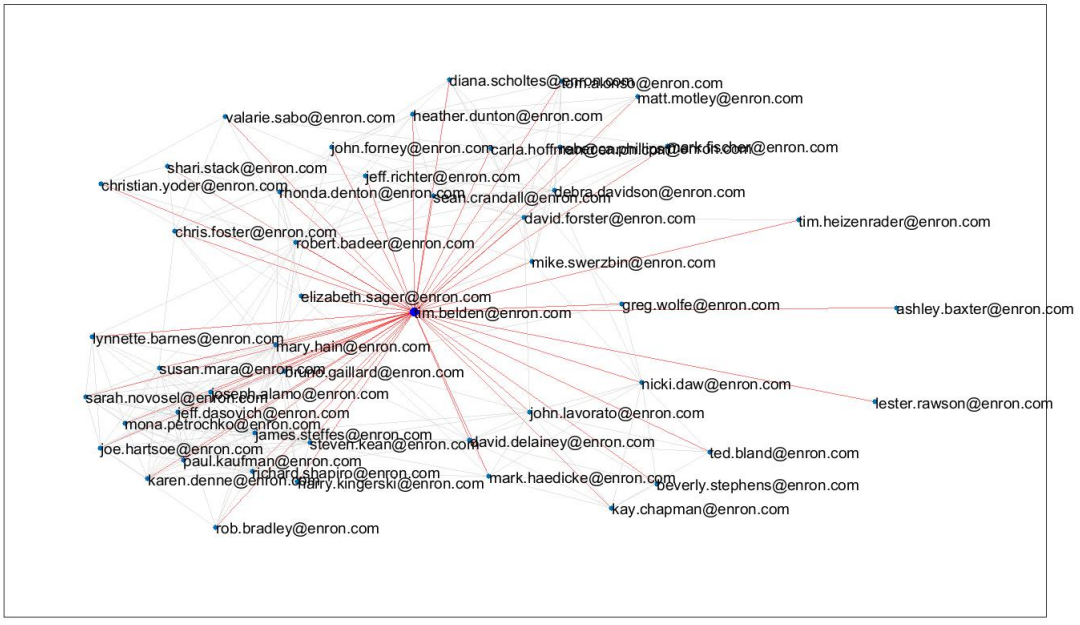


FIGURE 3.25: **Subgraph of the vertex corresponding to Timothy Beldon for August-2000**. Beldon is represented by the enlarged blue vertex in the centre, and the edges connected to it are highlighted in red.

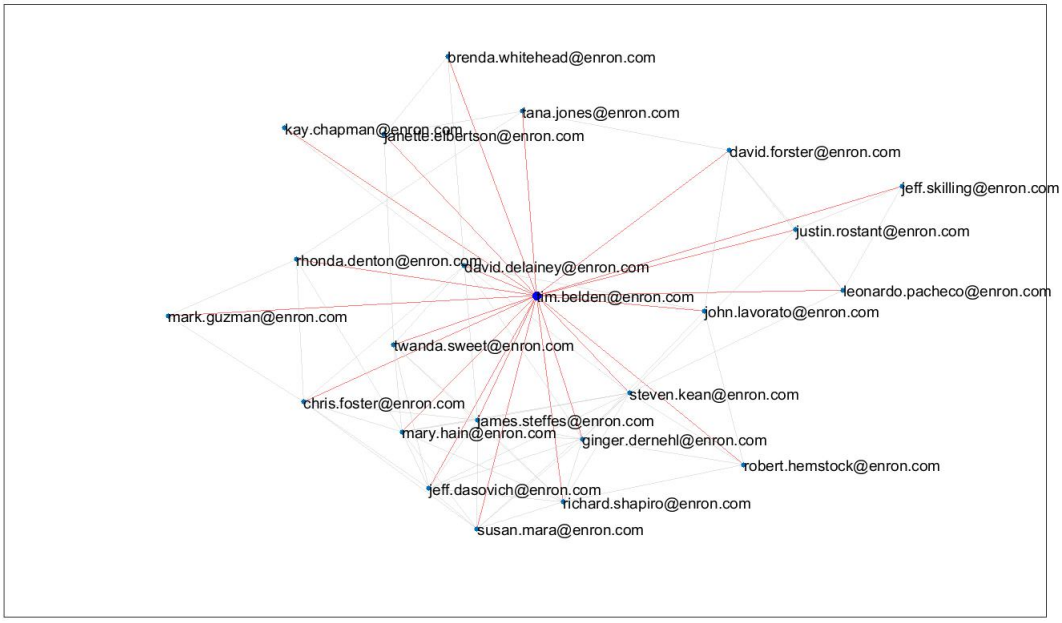


FIGURE 3.26: **Subgraph of the vertex corresponding to Timothy Beldon for September-2000**. Beldon is represented by the enlarged blue vertex in the centre, and the edges connected to it are highlighted in red.

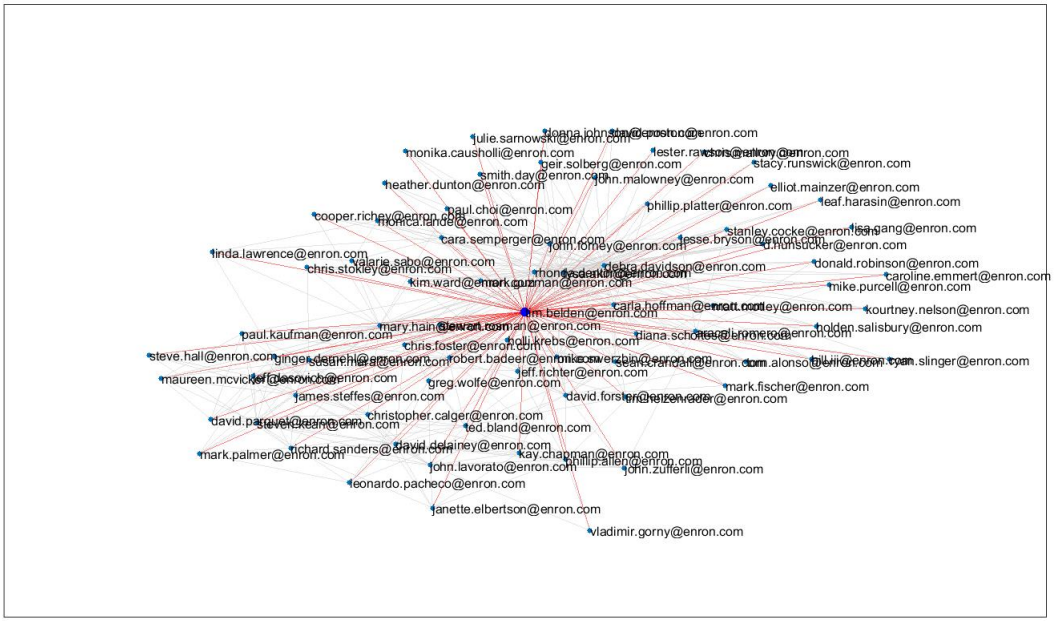


FIGURE 3.27: **Subgraph of the vertex corresponding to Timothy Beldon for October-2000**. Beldon is represented by the enlarged blue vertex in the centre, and the edges connected to it are highlighted in red.

observe how Fleming's connectivity patterns drastically change during this transition. CDP successfully detects this change at the corresponding time instant.

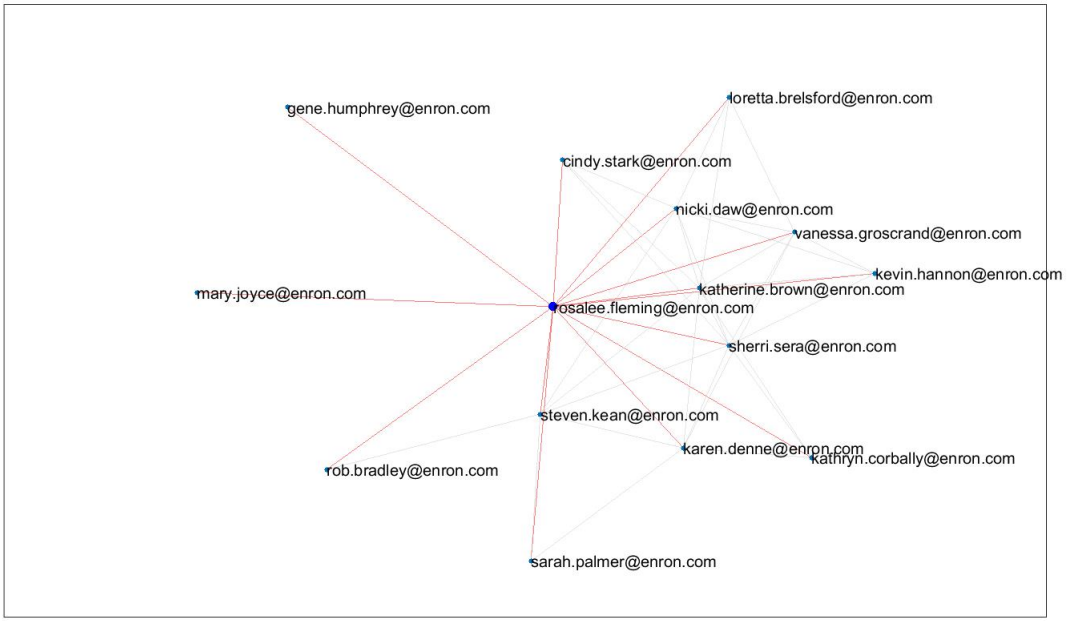


FIGURE 3.28: **Subgraph of the vertex corresponding to Rosalie Fleming for November-2000**. Fleming is represented by the enlarged blue vertex in the centre, and the edges connected to it are highlighted in red.

During the time instants between October-2001 to February-2002, CDP gives high change scores to employees in job roles such as risk analysts, senior specialists, presidents, vice presidents, and traders. This is justifiable as this was the time period where Enron's stocks started to fall, and bankruptcy was declared.

For the transition from September-2000 to October-2000, both ACT and ACTM give high change scores to Sara Shackleton (previous vice president to Enron North America). Figure 3.30 shows the histogram of the emails sent and received by Shackleton throughout this time period considered. We observe how Shackleton maintains a large number of overall communications, hence acts as a hub in the network. We observe a decrease in the degree at the time instant denoting October-2000 with respect to September-2000. Furthermore, our investigations show that there is approximately a 66% overlap in Shackleton's connections for these two time instants. Hence it is justifiable to assume that the change detected by ACT and ACTM simply reflects the change in degree for the corresponding vertex.

Christopher Calger (former executive in Enron's trading business) is also detected by ACT and ACTM for the transition from January-2001 to February-2001. However, when

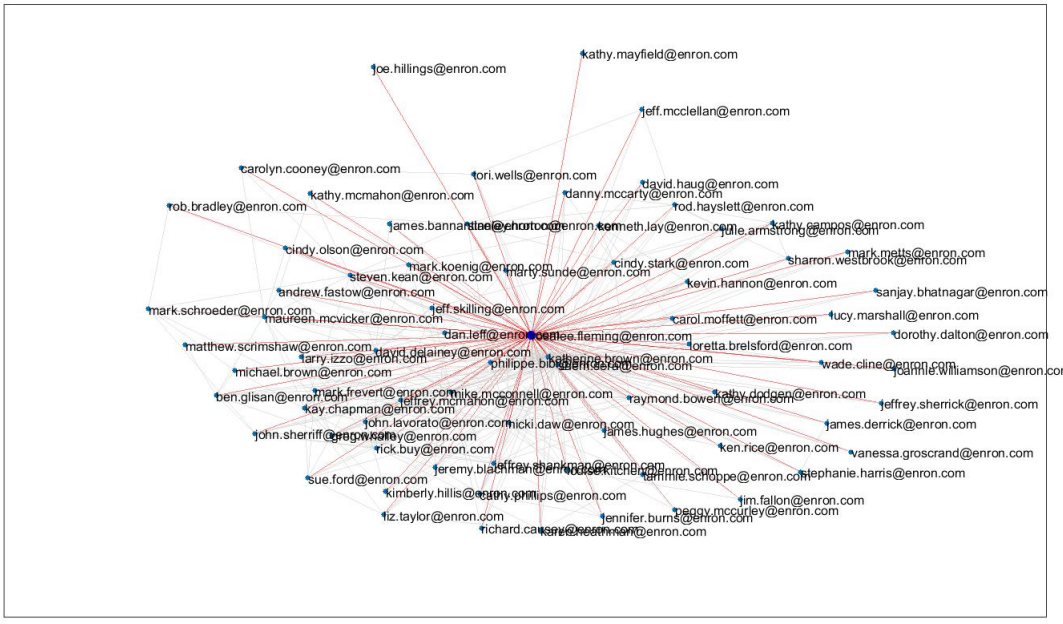


FIGURE 3.29: **Subgraph of the vertex corresponding to Rosalie Fleming for December-2000**. Fleming is represented by the enlarged blue vertex in the centre, and the edges connected to it are highlighted in red.

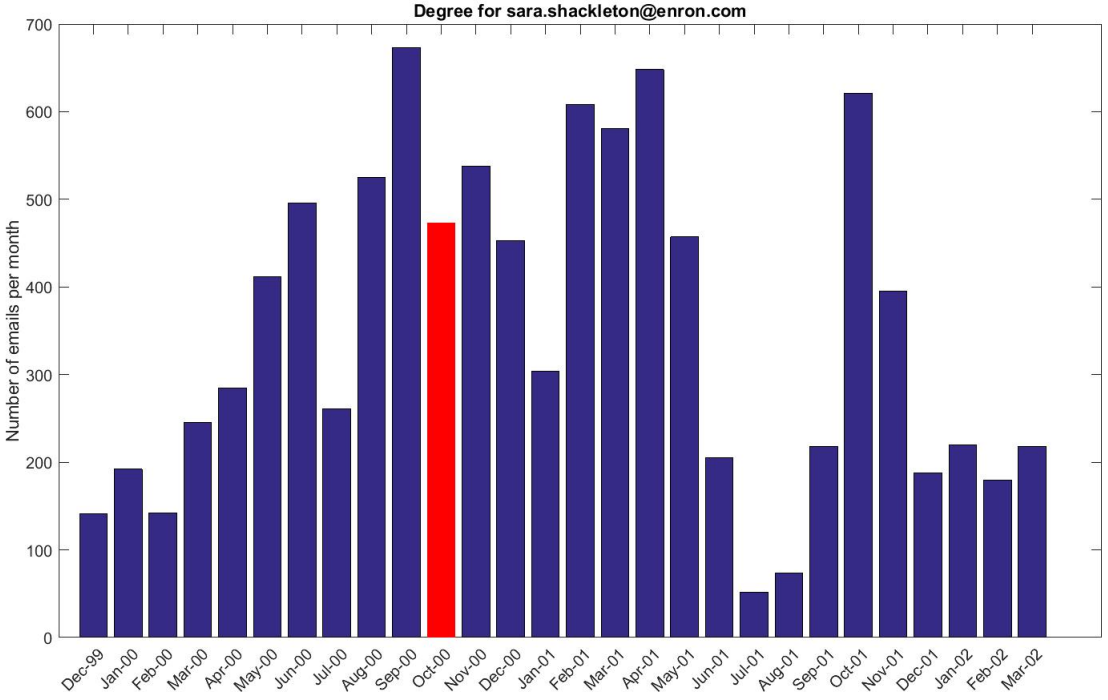


FIGURE 3.30: Number of emails sent and received by Sarah Shackleton during the 28 months

comparing the subgraph around the vertex corresponding to Calger in January-2001 (Figure 3.31) and February-2001 (Figure 3.32), we do not see a considerable change in his connections. Our further calculations show a 54% overlap in his connections. However, when observing the rate of change in the degree of the same vertex throughout the whole time period (Figure 3.33), February-2001 shows a slight increase.

In summary, CDP successfully detects some key players involved in the scandal as vertices which change the most during time instants corresponding to suspicious events in the Enron time-line. ACT and ACTM are focused mostly on change in the degree of the vertices, while CDP detects different types of changes.

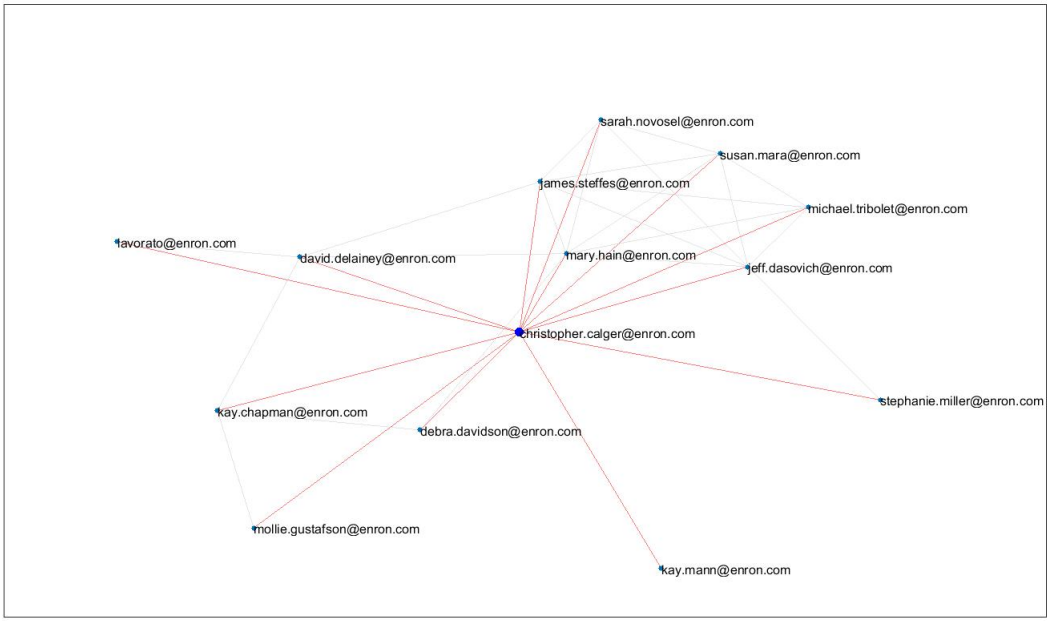


FIGURE 3.31: **Subgraph of the vertex corresponding to Christopher Calger for January-2001**. Calger is represented by the enlarged blue vertex in the centre and the edges connected to it are highlighted in red.

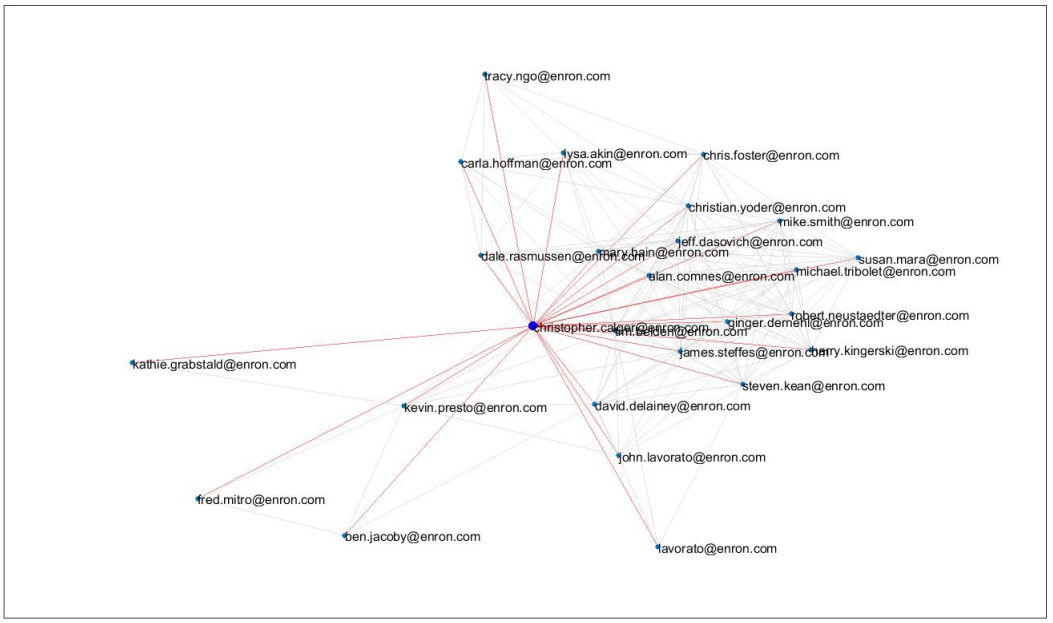


FIGURE 3.32: **Subgraph of the vertex corresponding to Christopher Calger for February-2001**. Calger is represented by the enlarged blue vertex in the centre and the edges connected to it are highlighted in red.

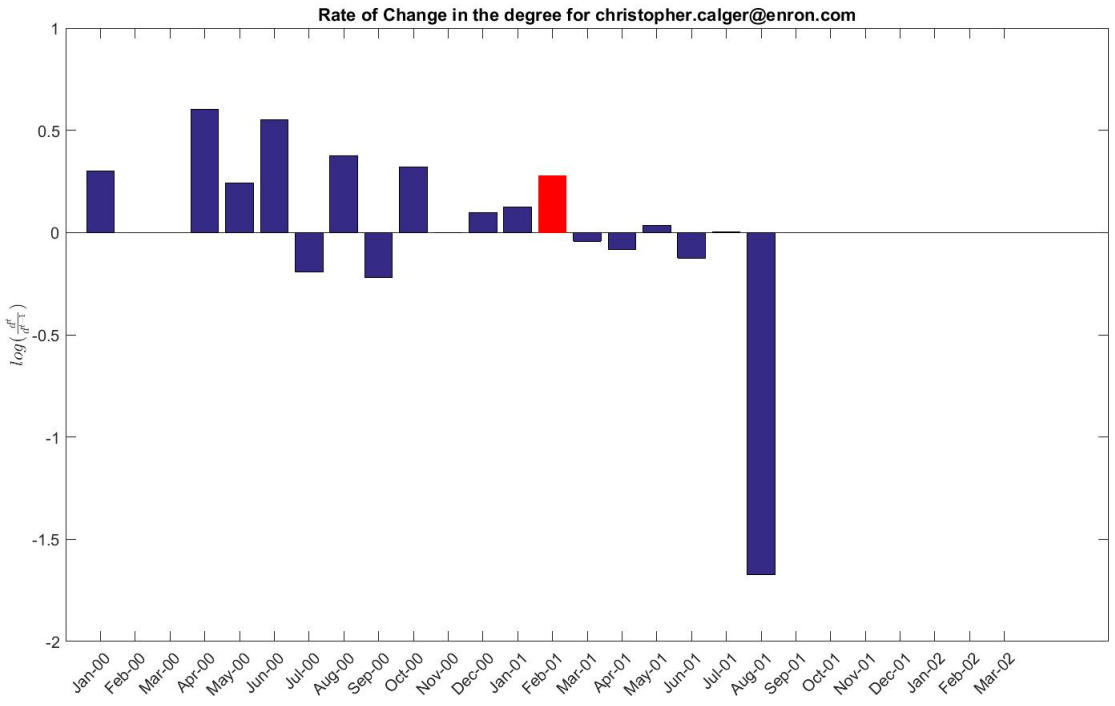


FIGURE 3.33: Rate of change in the number of emails sent and received by Christopher Calger during the 28 months

3.5 Summary

In this chapter, we present a novel method, CDP, to detect changes in vertex behaviour in a dynamic network represented as a time sequence of undirected and weighted graphs. We adopt a spectral embedding approach for this purpose.

Although there already exist change detection methods based on spectral embedding, such as ACT (Idé and Kashima, 2004), they are mainly designed to detect changes occurring in dense, well-connected graphs. Hence, when applied to sparse and heterogeneous graphs, they focus on the behaviour of highly active, dominant vertices. Changes occurring in vertices with moderate connectivity go unnoticed by these methods. Our approach adapts spectral techniques, commonly used in the area of spectral clustering, to obtain an embedding that represents all vertices in the graph. Our graph regularization method addresses sparseness and heterogeneity that are common in most real-world graphs. We apply a Procrustes analysis method to the embeddings to calculate change scores for vertices at each time instant. This is a novel application of Procrustes analysis. According to the results of our simulation experiments and experiments on the Enron email dataset, CDP successfully detects various changes involving vertices in a time evolving graph. These changes include changes in vertex degree, changes in community structure and unusual increases or decreases in edge weights. In all experiments, the performance of CDP was compared against two other spectral-based change detection methods, ACT and ACTM, which did not address sparsity and heterogeneity issues in the embedding stage. These baseline methods failed to detect the changes in the majority of experiments.

To conclude, this chapter presents a novel change detection method combining spectral embedding and Procrustes analysis techniques. Our method successfully detects a wide range of vertex-based changes that closely relate to changes occurring in most real-world dynamic networks.

Chapter 4

Change Detection in Multi-view Dynamic Networks using Procrustes Analysis

In many situations we have a multi-view network encoding multiple types of relationships between a set of entities. For example, a network of university friends may also be connected as friends on Facebook, as room-mates in hostels, and contacts in LinkedIn. Additionally, a multi-view network may arise from different attributes of a single relationship. For example, a phone-call between two people on a particular day has attributes such as call duration, number of times connected, and so forth. In Chapter 3, we outlined our novel change detection method, called CDP, for a single-view network. CDP employs a matrix-based spectral embedding procedure to perform change detection. In this chapter, we extend the CDP method to handle multi-view networks represented as three-dimensional tensors.

The multiple relationships in a multi-view network can be conceptualized as a multi-graph, which is a graph with multiple edges between pairs of vertices. One way to represent a multi-graph is by partitioning the edge set using the edge attributes and creating a number of different graphs on the same vertex set. We represent a dynamic multi-view network as a time sequence of multi-graphs. From Chapter 3, one of the most crucial steps in our change detection procedure is the spectral embedding step at each time instant. It is challenging to obtain an embedding from a multi-graph because of the multiple edge attributes. An embedding from a multi-graph should fully exploit the connectivity structure represented by each edge attribute and also combine the common structure represented across different edge attributes in a meaningful manner. Different edge attributes may depict different structures. For example, one edge attribute

may depict a clear block structure while another may depict a very sparse or a noisy structure. Our focus in this chapter is to propose a multi-graph embedding method that can effectively combine information from different edge attributes in order to strengthen the performance of change detection.

We represent a multi-graph as a three-dimensional tensor and propose two approaches to extract an embedding from this tensor. By employing these embedding techniques, we extend our CDP method in two ways creating two methods: MCDP–I and MCDP–II (MCDP-multi-view change detection using Procrustes analysis) which detect changes involving vertices in a time sequence of tensors. We apply MCDP-I and MCDP-II to both simulated and real-world datasets and evaluate the performance of change detection. In our simulations, we study change detection performance in four situations ranging from simple to most difficult: (i) the multi-graph has the same block structure across graphs, and vertices undergo the same change in all graphs; (ii) the multi-graph has the same block structure across graphs and vertices undergo the same change in all graphs, but some graphs are noisy and have many inter-block edges; (iii) vertices undergo the same change in all graphs, but the multi-graph has a different block structure across graphs; and (iv) the multi-graph has a different block structure, and vertices undergo different changes across graphs. Based on the results of our experiments, both MCDP-I and MCDP-II show good change detection performance in all cases, but with MCDP-II performing better than MCDP-I in the most difficult situations.

This chapter is organized as follows. In Section 4.1, we provide a detailed description of our multi-view change detection framework. We introduce our multi-graph embedding method I (Section 4.1.3.1) and method II (Section 4.1.3.2), and extend CDP to MCDP-I and MCDP-II (Section 4.1.3.3) to detect vertex-based changes in a multi-view dynamic network. The change detection performance of MCDP-I and MCDP-II is then evaluated by performing several simulation experiments (Section 4.2) as well as a real-world application (Section 4.3). For simulation experiments, the performance of MCDP-I and MCDP-II is compared against several base-line methods that employ existing multi-graph embedding techniques (detailed in Section 4.2.1). Finally we summarize our findings in Section 4.4.

4.1 Problem Framework

4.1.1 Notation and Terminology

Let $\mathbf{G}^1, \mathbf{G}^2, \ldots, \mathbf{G}^{\mathcal{T}}$ be a sequence of multi-graphs defined over time instants, $t = 1, 2, \ldots \mathcal{T}$. Each $\mathbf{G}^t$ consists of a set of l graphs, $\{G_1^t, G_2^t, \ldots, G_l^t\}$, on the same set

of vertices. More formally, $\mathbf{G}^t = \{G_k^t\}_{k=1}^l$, where each graph, G_k^t, consists of the set of n vertices, $V = \{v_1, \ldots v_n\}$, and a set of edges, $E_k^t \leq n^2$. Let W_k^t be the weighted symmetric adjacency matrix of dimension $n \times n$, where each element, $W_{i,j,k}^t \geq 0$, denotes the weight of the edge between vertices i and j in G_k^t. Each $\mathbf{G}^t$ is represented using a weighted adjacency tensor, $\mathbf{W}^t$, where the mode-3 slices (Figure 2.16 in Section 2.5.2.1) are the matrices, $W_1^t, W_2^t, \ldots, W_l^t$.

Let us define some transformations applied to each W_k^t in this Chapter[1]:

- Let $\ddot{W}_k$ be the $n \times n$ *log transformed weighted adjacency matrix* corresponding to G_k, where each element, $\ddot{W}_{i,j,k}$, is given by

$$\ddot{W}_{i,j,k} = \log_{10}(W_{i,j,k} + 1) \quad \forall i, j \in \{1, \ldots, n\}. \tag{4.1}$$

- Let $\grave{W}_k$ be the $n \times n$ *scaled weighted adjacency matrix* corresponding to G_k, where each element, $\grave{W}_{i,j,k}$ is given by

$$\grave{W}_{i,j,k} = \frac{\ddot{W}_{i,j,k}}{\max_{i,j}\{\ddot{W}_{i,j,k}\}}. \tag{4.2}$$

- Let M_k be the $n \times n$ *regularized degree normalized weighted adjacency matrix* corresponding to G_k given by

$$M_k = D_{(k,\tau)}^{-1/2} W_{(k,\tau)} D_{(k,\tau)}^{-1/2}, \tag{4.3}$$

where

$$W_{(k,\tau)} = \grave{W}_k + \tau_k \mathbf{1}\mathbf{1}^T, \tag{4.4}$$

where $\mathbf{1}$ is an n-dimensional column vector containing all ones and $D_{(k,\tau)}$ is the $n \times n$ degree matrix for $W_{(k,\tau)}$. The regularizer, τ_k, is calculated as

$$\tau_k = \frac{1}{4n^2} \sum_{i,j} \grave{W}_{i,j,k}. \tag{4.5}$$

4.1.2 Problem Statement

Our goal is to calculate a change score for each vertex in $\mathbf{G}^t$ with respect to the recent past behaviour. We follow Definition 3.1 in Chapter 3 to define the change score for v_i at time instant t. According to this definition, our overall change detection procedure can be summarized as follows:

[1]For notational convenience, we drop the superscript t and use the notation W_k.

1. Obtain a feature, $\mathbf{x}_i^t$, for v_i from each $\mathbf{G}^t$, where $\mathbf{x}_i^t \in \mathbb{R}^{d^t}$ and $d^t \in \mathbb{Z}_+$.
2. Obtain a profile feature, $\bar{\mathbf{x}}_i^{t-1}$, for v_i from the previous w time instants, $\mathbf{G}^{t-w}, \ldots, \mathbf{G}^{t-1}$, where $\bar{\mathbf{x}}_i^{t-1} \in \mathbb{R}^{\bar{d}^{t-1}}$ and $\bar{d}^{t-1} \in \mathbb{Z}_+$.
3. Calculate the dissimilarity between $\mathbf{x}_i^t$ and $\bar{\mathbf{x}}_i^{t-1}$ using function f, and obtain the change scores, z_i^t, for $v_i \in V$.

In Sections 3.2.3, 3.2.4 and 3.2.5, we explained how the above three steps are implemented when a graph is represented as a weighted adjacency matrix. When we represent a multi-graph as a tensor, the same spectral embedding procedure discussed in Section 3.2.3 cannot be applied to extract features for vertices. In Section 4.1.3, we introduce two procedures to extract an embedding from a tensor. Thereafter, we use the same techniques discussed in Sections 3.2.4 and 3.2.5 for profile feature extraction and change score calculation, respectively.

4.1.3 The Change Detection Method

Let us first recall our vertex-based feature extraction procedure for a single graph, G_1, from Section 3.2.3. At each time instant, we calculate the embedding, $X \in \mathbb{R}^{n \times d}$, of M_1 by solving

$$\max_X \parallel X^T M_1 X \parallel_F^2, \tag{4.6}$$

subject to $X^T X = I$. We estimate X by performing SVD on M_1 and extracting d principal singular vectors. The truncation dimension d is determined using Algorithm 1 in Section 3.2.3. Each row, $\mathbf{x}_i$, is the feature of v_i at a given time instant.

In the presence of multiple graphs, at each time instant, we construct the representation tensor, $\mathbf{M}$, where mode-3 slices are the set of representation matrices, $\{M_1, M_2, \ldots, M_l\}$, corresponding to graphs, $G_1, G_2, \ldots, G_l$, respectively. As we now have multiple representation matrices, Equation 4.6 is no longer applicable. In the next two sections, we will discuss two methods to obtain an embedding from the representation tensor.

4.1.3.1 Method I

Our goal is to embed the vertices into a single d-dimensional latent Euclidean space by jointly considering all slices in $\mathbf{M}$. Motivated by the multi-graph spectral embedding procedure in Liu et al. (2013, section 3.2.2), we can extend objective function 4.6 to multiple slices giving

$$\max_X \sum_{k=1}^{l} \parallel X^T M_k X \parallel_F^2, \tag{4.7}$$

where $X^T X = I$. An approximate solution to expression 4.7 can be obtained through the higher order singular value decomposition (HOSVD).

4.1.3.1.1 Brief Review of HOSVD

HOSVD computes the Tucker decomposition of a three mode rank- (r_1, r_2, r_3) tensor, $\mathbf{A} \in \mathbb{R}^{i_1 \times i_2 \times i_3}$, finding that

$$\mathbf{A} = \mathbf{S} \times_1 U \times_2 V \times_3 W,$$

where $U \in \mathbb{R}^{i_1 \times r_1}$, $V \in \mathbb{R}^{i_2 \times r_2}$, and $W \in \mathbb{R}^{i_3 \times r_3}$ are the orthonormal factor matrices, and the core tensor, $\mathbf{S} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, satisfies *all-orthogonality* and *ordering* constraints (for a more detailed description, we refer the reader to Section 2.5.2.3). The low-rank approximation of $\mathbf{A}$ is obtained by solving the least squares cost function

$$\min_{U,V,W,\mathbf{S}} \parallel \mathbf{A} - \mathbf{S} \times_1 U \times_2 V \times_3 W \parallel_F^2, \tag{4.8}$$

where the solutions, $U \in \mathbb{R}^{i_1 \times d_1}$, $V \in \mathbb{R}^{i_2 \times d_2}$ and $W \in \mathbb{R}^{i_3 \times d_3}$, are column-wise orthonormal with $d_1 < r_1$, $d_2 < r_2$, $d_3 < r_3$, and $\mathbf{S} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$. De Lathauwer et al. (2000a) showed that the minimization problem in expression 4.8 is equivalent to the maximization of the expression

$$\max_{U,V,W} \parallel \mathbf{A} \times_1 U^T \times_2 V^T \times_3 W^T \parallel_F^2, \tag{4.9}$$

where, $\mathbf{S} = \mathbf{A} \times_1 U^T \times_2 V^T \times_3 W^T$. HOSVD obtains an approximation to problem in expression 4.9 by computing U, V and W as the principal left singular vectors of mode–1 matricization, $\mathbf{A}_{(1)} \in \mathbb{R}^{i_1 \times i_2 i_3}$, mode–2 matricization, $\mathbf{A}_{(2)} \in \mathbb{R}^{i_2 \times i_3 i_1}$, and mode–3 matricization, $\mathbf{A}_{(3)} \in \mathbb{R}^{i_3 \times i_2 i_1}$ of the tensor, respectively.

If we set the factor matrix of mode–3 to be the identity matrix with dimensions $i_3 \times i_3$, the maximization problem in expression 4.9 can be written as

$$\max_{U,V} \parallel \mathbf{A} \times_1 U^T \times_2 V^T \times_3 I \parallel_F^2 . \tag{4.10}$$

If tensor $\mathbf{A}$ is partially symmetric, that is, if each slice $A_{:,:,k}$ is a symmetric matrix, expression 4.10 further reduces to

$$\max_{U} \parallel \mathbf{A} \times_1 U^T \times_2 U^T \times_3 I \parallel_F^2 . \tag{4.11}$$

Due to the partial symmetry of $\mathbf{A}$, the principal left singular vectors of $\mathbf{A}_{(1)}$ and $\mathbf{A}_{(2)}$ are the same. We direct the interested reader to Section 2.5.2 for a detailed review of Tensors.

4.1.3.1.2 Multi-graph Embedding using HOSVD

Let us now return to expression 4.7. We have a multi-graph with undirected edges, encoded as a partially symmetric three mode tensor, $\mathbf{M}$, of dimensions $n \times n \times l$. Thus, expression 4.7 can also be written as

$$\max_{X} \parallel \mathbf{M} \times_1 X^T \times_2 X^T \times_3 I \parallel_F^2, \tag{4.12}$$

where $X \in \mathbb{R}^{n \times d}$ has orthonormal columns. With comparison to expression 4.11, an approximate solution to expression 4.12, is the principal d left singular vectors of $\mathbf{M}_{(1)}$ (Figure 4.1). The truncation dimension d can be estimated using Algorithm 1.

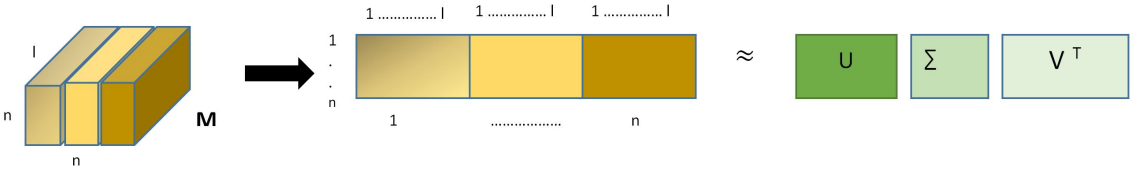


FIGURE 4.1: **Illustration of tensor decomposition**. The representation tensor of a multi-graph is three-dimensional. The mode–1 (as well as mode–2 since the network is undirected) fibers represent the vertices of the network. We apply SVD on the mode–1 matricized tensor and extract the principal left singular vectors as the embedding of the multi-graph.

Algorithm 4 presents a step by step description of our tensor-based spectral embedding method-I. In Section 3.2.3, we employed matrix decomposition to obtain an embeddding from a single-view network. In this section, we employ tensor decomposition to obtain an embedding from a multi-view network. By implementing Algorithm 4, each multi-graph, $\mathbf{G}^t$, in the sequence is represented as a low dimensional embedding, $X^t \in \mathbb{R}^{n \times d^t}$.

4.1.3.2 Method II

Another method to obtain a single embedding from $\mathbf{M}$ is to initially obtain embeddings separately from $M_1, \ldots, M_l$, and then combine them into one embedding. From Equation 4.6, an embedding from each M_k is an approximate solution to

$$\max_{X_k} \parallel X_k^T M_k X_k \parallel_F^2 \tag{4.13}$$

Algorithm 4 Tensor-based Spectral Embedding Procedure–I

Input: Tensor, $\mathbf{W}$, of dimensions $n \times n \times l$, where the mode-3 slices are the weighted adjacency matrices, $W_1, W_2, \ldots W_l$.

Output: Final embedding, $X \in \mathbb{R}^{n \times d}$

1: **for** $k = 1$ **to** l **do**
2: Update: $W_k = \log_{10}(W_k + \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)$
3: Update: $W_k = \frac{W_k}{\max_{i,j}\{W_{i,j,k}\}}$
4: Calculate $\tau_k = \frac{1}{4n^2}\sum_{i,j} W_{i,j,k}$
Update: $W_{(k,\tau)} = W_k + \tau_k \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T$
Calculate $D_{(k,\tau)}$ with diagonal elements, $[D_{(k,\tau)}]_{i,i} = \sum_{j=1}^{n}[W_{(k,\tau)}]_{i,j}$
Calculate $M_k = D_{(k,\tau)}^{-1/2} W_{(k,\tau)} D_{(k,\tau)}^{-1/2}$
5: **end for**
6: Build representation tensor, $\mathbf{M}$, by stacking $M_1, \ldots M_l$ along the third dimension
7: Obtain mode–1 matricization $\mathbf{M}_{(1)}$
8: Perform SVD: $\mathbf{M}_{(1)} = U\Sigma V^T$
9: Estimate d by applying Algorithm 1 on $\mathbf{M}_{(1)}$
10: Obtain the final embedding, $X \in \mathbb{R}^{n\times d}$, where the columns are the d left singular vectors, $[\mathbf{u}_2, \ldots, \mathbf{u}_{d+1}]$

subject to $X_k^T X_k = I$ and $X_k \in \mathbb{R}^{n\times d_k}$. Using the resulting embeddings, $X_1, \ldots, X_l$, an average can be calculated to obtain a single embedding. However, this task is not straightforward for two reasons.

1. The dimensions, $d_1, \ldots, d_l$, can be different, so averaging $X_1, \ldots, X_l$ is not possible.

2. The embeddings, $X_1, \ldots, X_l$, are unique up to Euclidean similarity transformations such as scale, rotation, and reflection, since each X_k is obtained by performing SVD on M_k and extracting d_k principal left singular vectors. Thus, averaging X_k directly, may not give a meaningful result.

In Section 3.2.4, we faced the same challenges as mentioned above when calculating the profile embedding by averaging the sequence of embeddings from the recent past time instants. In order to overcome those challenges, we formulated a generalized orthogonal Procrustes analysis (GPA) technique. In this Section, we use that same technique to obtain an average embedding from $X_1, \ldots, X_l$. Algorithm 5 presents a detailed description of our second spectral embedding procedure. A further illustration is given in Figure 4.2. MCDP-II is inspired by the method proposed in Tang et al. (2012), where a generalized canonical correlation analysis procedure (CCA) is employed to combine the embeddings from different views. Later in Section 4.2.1, we discuss Tang et al. (2012)'s method under comparison methods.

We perform the multi-graph spectral embedding procedure given in Algorithm 5 at each time instant to reduce each multi-graph, $\mathbf{G}^t$, in the sequence into a low dimensional embedding, $X^t \in \mathbb{R}^{n\times d^t}$.

Algorithm 5 Tensor-based Spectral Embedding Procedure–II

Input: Tensor, $\mathbf{W}$, of dimensions $n \times n \times l$, where the mode-3 slices are the weighted adjacency matrices, $W_1, W_2, \ldots W_l$.

Output: Embedding, $X \in \mathbb{R}^{n \times d}$, of the multi-graph

for $k = 1$ **to** l **do**
  Update: $W_k = \log_{10}(W_k + \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)$
  Update: $W_k = \frac{W_k}{\max_{i,j}\{W_{i,j,k}\}}$
  Calculate $\tau_k = \frac{1}{4n^2}\sum_{i,j} W_{i,j,k}$
  Update: $W_{(k,\tau)} = W_k + \tau_k \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T$
  Calculate $D_{(k,\tau)}$ with diagonal elements, $[D_{(k,\tau)}]_{i,i} = \sum_{j=1}^{n}[W_{(k,\tau)}]_{i,j}$
  Calculate $M_k = D_{(k,\tau)}^{-1/2} W_{(k,\tau)} D_{(k,\tau)}^{-1/2}$
  Perform SVD: $M_k = U_k \Sigma_k U_k^T$
  Estimate d_k by applying Algorithm 1 on M_k
  Obtain $n \times d_k$ matrix, X_k, where the columns are the d_k principal singular vectors in U_k starting from the second vector onwards
end for
Calculate $d_{\max} = \max\{d_1, \ldots, d_l\}$.
for $k = 1, \ldots, l$ **do**
  if $d_k < d_{\max}$ **then**
    append $d_{\max} - d_k$ columns of zeros to X_k
  end if
end for
Input $X_1, \ldots, X_l$ into Algorithm 2 and obtain the combined mean embedding, X

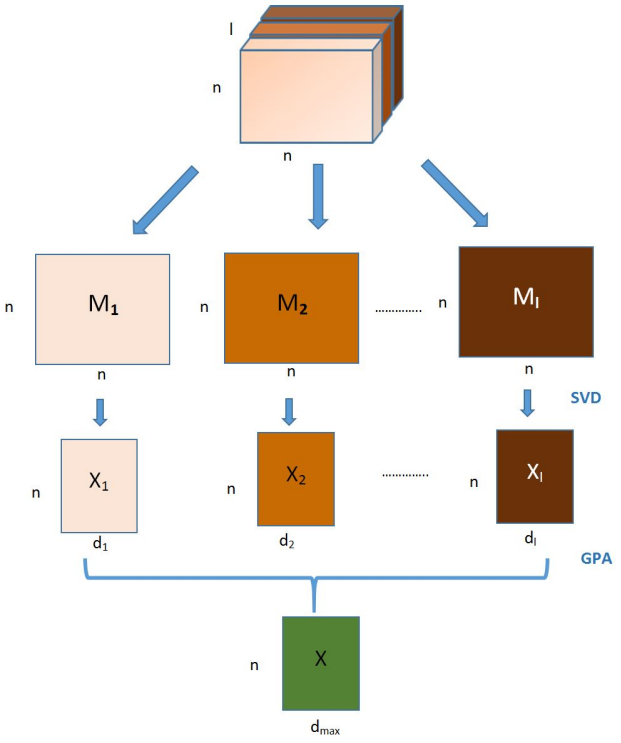


FIGURE 4.2: **Illustration of multi-graph embedding method-II**. First each representation matrix is separately decomposed using SVD. After that, the embeddings from each slice are averaged to obtain a single embedding of the multi-graph using our GPA procedure.

4.1.3.3 Proposed Algorithm - MCDP (Multi-graph Change Detection using Procrustes Analysis)

In Sections 4.1.3.1 and 4.1.3.2, we proposed two methods to obtain an embedding from a tensor. By using either one of these methods, the sequence of multi-graphs, $\mathbf{G}_1, \ldots, \mathbf{G}^{\mathcal{T}}$, is reduced to a sequence of low dimensional embeddings, $X^1, \ldots, X^{\mathcal{T}}$. We can then apply the same moving window based approach discussed in Sections 3.2.4 and 3.2.5 to calculate the profile embedding and the change scores, respectively. In this section, we present our change detection algorithm, MCDP, given in Algorithm 6. When the spectral embedding method in Section 4.1.3.1 is applied, it is called the MCDP–I algorithm, and when the spectral embedding method in Section 4.1.3.2 is applied, it is called the MCDP–II algorithm.

Algorithm 6 Multi-view Change Detection using Procrustes Analysis - MCDP-I and MCDP-II

Input: (i) Time sequence of tensors, $\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^{\mathcal{T}}$, where each $\mathbf{W}^t$ has dimensions $n \times n \times l$ and mode-3 slices are the set of matrices, $\{W_k^t\}_{k=1}^{l}$ (ii) window size w

Output: Time sequence of vertex change scores, $\mathbf{z}^1, \mathbf{z}^2, \ldots \mathbf{z}^{\mathcal{T}}$. Each $\mathbf{z}^t$ is a vector of dimension n

for $t = 1$ **to** $\mathcal{T}$ **do**
  Input $\mathbf{W}^t$ to Algorithm 4 or 5 to obtain embedding, $X^t \in \mathbb{R}^{n \times d^t}$
end for
for $t = w + 1$ **to** $\mathcal{T}$ **do**
  Let $d_{\max_1} = \max\{d^{t-w}, \ldots, d^{t-1}\}$
  for $t' \in \{t - w, \ldots, t - 1\}$ **do**
    if $d^{t'} < d_{\max_1}$ **then**
      append $d_{\max_1} - d^{t'}$ columns of zeros to $X^{t'}$
    end if
  end for
  Input $X^{t-w}, \ldots, X^{t-1}$ into Algorithm 2 to obtain the profile embedding, $\overline{X}^{t-1}$
  Let $d_{\max_2} = \max\{d_{\max_1}, d^t\}$
  if $d_{\max_1} < d_{\max_2}$ **then**
    append $d_{\max_2} - d_{\max_1}$ columns of zeros to $\overline{X}^{t-1}$
  end if
  if $d^t < d_{\max_2}$ **then**
    append $d_{\max_2} - d^t$ columns of zeros to X^t
  end if
  Align X^t and $\overline{X}^{t-1}$ with each other using Algorithm 2, and obtain the adjusted embeddings, $\widehat{X}^t$ and $\widehat{\overline{X}}^{t-1}$, and the mean $\hat{\mu}^t$
  Calculate vertex change scores, $z_i^t = \frac{\|\widehat{X}^t_{i,.} - \widehat{\overline{X}}^{t-1}_{i,.}\|_F^2}{\|\hat{\mu}^t\|_F}$
end for

We evaluate the performance of MCDP–I and MCDP-II by conducting simulation experiments and applying them to real-world datasets.

4.2 Simulation Experiments

In Chapter 3, we conducted experiments on nine change scenarios to a single-view dynamic network. In this section, we extend some of those change scenarios for a multi-view dynamic network. For each scenario, we generate a time sequence of tensors, $\mathbf{W}^1, \mathbf{W}^2, \ldots \mathbf{W}^{\mathcal{T}}$, to represent a dynamic multi-view network. We assume that the edges of the multi-graphs have distribution F_0 and when a change occurs, the distribution becomes F_1. Here we only consider a change-point, t^*:

$$\mathbf{W}^t \sim \begin{cases} F_1 & \text{if } t = t^*, \\ F_0 & \text{otherwise .} \end{cases} \tag{4.14}$$

where $w < t^* \leq \mathcal{T}$.

We generate each W_k^t using the DCSBM model employed in Section 3.3.2. We set the number of mode–3 slices, $l = 4$. Each change scenario corresponds to a transition from generative model F_0 to F_1 at the changed time instant. All change scenarios are evaluated using three main situations ranging from simple to most difficult:

1. *Simple situation*: All the four slices contain the same clear three block structure representing a multi-graph with a common community structure across graphs (Table 4.1).

2. *Noisy situation*: All the four slices contain the same clear three block structure, but some slices contain a large number of inter-block elements. This represents a multi-graph with a common community structure across graphs, but with some graphs containing a large number of inter-block edges (Table 4.2).

3. *Complex situation*: The block structure varies significantly among the four slices representing a multi-graph with different connectivity structure across graphs (Table 4.3).

For each change scenario, we generate a sequence of 20 tensors, that is, we set $\mathcal{T} = 20$. A change point is injected at $t^* = 11$ (Equation 4.14). We use a window, w of length five and calculate change scores for vertices at each time instant. We repeat this 100 times and calculate the same performance measures described in Section 3.3.4. That is, for each change scenario, we calculate both $\boldsymbol{\eta}^t$ (Equation 3.18) and $\bar{\boldsymbol{\eta}}^t$ (Equation 3.19) to measure change detection performance.

TABLE 4.1: **Parameter settings for F_0 and F_1 for different change scenarios in the simple situation.** The parameter values are as initially specified in Table 3.1 in Chapter 3, unless explicitly stated.

		F_0	F_1		
			group-change	hetero-to-homo	fragment
Model		$\mathcal{M}_1$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_3$
Slice	1	$\alpha = 0.01, \beta = 0.02, \gamma = 0.03$			
	2	$\alpha = 0.03, \beta = 0.01, \gamma = 0.02$			
	3	$\alpha = 0.02, \beta = 0.03, \gamma = 0.01$			
	4	$\alpha = 0.03, \beta = 0.03, \gamma = 0.03$			
Figure		D.1	D.2	D.3	D.4
Vertices changed			$\{1 \dots 600\}$	$\{1 \dots 300\}$	$\{601 \dots 900\}$

TABLE 4.2: **Parameter settings for F_0 and F_1 for different change scenarios in the noisy situation.** The parameter values are as initially specified in Table 3.1 in Chapter 3, unless explicitly stated.

		F_0	F_1		
			group-change	hetero-to-homo	fragment
Model		$\mathcal{M}_1$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_3$
Slice	1	$\alpha = 0.01, \beta = 0.02, \gamma = 0.03, \lambda = 0.8$			
	2	$\alpha = 0.03, \beta = 0.01, \gamma = 0.02, \lambda = 0.8$			
	3	$\alpha = 0.02, \beta = 0.03, \gamma = 0.01, \lambda = 0.5$			
	4	$\alpha = 0.03, \beta = 0.03, \gamma = 0.03, \lambda = 0.3$			
Figure		D.5	D.6	D.7	D.8
Vertices changed			$\{1 \dots 600\}$	$\{1 \dots 300\}$	$\{601 \dots 900\}$

TABLE 4.3: **Parameter settings for F_0 and F_1 for different change scenarios in the complex situation.** The inter-block probabilities for each slice are as given in Table 4.1 unless explicitly stated. Other parameter values not specified are default values from the model specifications in Table 3.1 in Chapter 3.

Slice	F_0	F_1			
		group-change	hetero-to-homo	fragment	mixed
1	$\mathcal{M}_1$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_3$	$\mathcal{M}_5$
2	$\mathcal{M}_2$	$\mathcal{M}_2$ $\mathbf{g} = [150, 75, 375, 300]$	$\mathcal{M}_2$ $\boldsymbol{\theta}_{1,\dots,300} = \mathbf{c}$	$\mathcal{M}_2$ $\gamma = 0.002$	$\mathcal{M}_2$ $\boldsymbol{\theta}_{1,\dots,300} = \mathbf{c}$
3	$\mathcal{M}_6$	$\mathcal{M}_6$ $\mathbf{g} = [150, 450, 300]$	$\mathcal{M}_6$ $\boldsymbol{\theta}_{1,\dots,300} = \mathbf{c}$	$\mathcal{M}_5$ $\gamma = 0.001$	$\mathcal{M}_5$
4	$\mathcal{M}_1$ $\beta = 0.0015$	$\mathcal{M}_4$ $\beta = 0.0015$ $\mathbf{g} = [150, 450, 300]$	$\mathcal{M}_5$ $\beta = 0.0015$ $\boldsymbol{\theta}_{1,\dots,300} = \mathbf{c}$	$\mathcal{M}_5$ $\beta = 0.0010$ $\gamma = 0.0030$	$\mathcal{M}_1$
Figure	D.9	D.10	D.11	D.12	D.13
Vertices changed		$\{1 \dots 600\}$	$\{1 \dots 300\}$	$\{601 \dots 900\}$	$\{1 \dots 300\}$ [a] $\{301 \dots 600\}$ [b]

[a] This set of vertices change behaviour in all slices, hence we refer to them as *All*.
[b] This set of vertices change behaviour only in two slices, hence we refer to them as *Two*.

4.2.1 Comparison Methods

We compare MCDP-I and MCDP-II with six other methods. Each of these methods provides an alternative strategy to calculate an embedding from the four slices of the multi-graph at each time instant. The change detection performance of each method is evaluated using the same measures, $\boldsymbol{\eta}^t$ (Equation 3.18) and $\bar{\boldsymbol{\eta}}^t$ (Equation 3.19), described in Section 3.3.4.

1. **AVG**
   A natural way to combine information in multiple slices of the tensor, $\mathbf{W}^t$, is to obtain an average weighted adjacency matrix, $\overline{W}^t$ (Han et al., 2015). Specifically, we calculate
   $$\overline{W}^t = \frac{1}{l}\sum_{k=1}^{l} W_k^t, \tag{4.15}$$
   where $\overline{W}^t$ has dimension $n \times n$. Once $\overline{W}^t$ is calculated, we can employ the same matrix-based spectral embedding procedure discussed in Section 3.2.3 to obtain a low dimensional embedding.

2. **MAX**
   The MAX method is a variant of the AVG method to combine the four slices in $\mathbf{W}^t$. Given $\mathbf{W}^t$, we construct a symmetric, weighted adjacency matrix, $W_{\max}^t$, of dimension $n \times n$, where $[W_{\max}]_{i,j}^t = \max_l\{W_{i,j,l}^t\}$. Once $W_{\max}^t$ is calculated, we can employ the same matrix based spectral embedding procedure discussed in Section 3.2.3 to obtain a low dimensional embedding.

3. **AVGU**
   The averaged utility (AVGU) method is introduced in Tang et al. (2009). The term *utility* refers to what we call the *representation matrix* in this thesis. Recall from Section 4.1.1 that we use $\mathbf{M}^t$ as the representation tensor of each $\mathbf{G}^t$. An average utility matrix, $\overline{M}^t$, is given by
   $$\overline{M}^t = \frac{1}{l}\sum_{k=1}^{l} M_k^t.$$
   Applying the ideas in Tang et al. (2012), an embedding, X, from $\overline{M}$ can be seen as an approximate solution to the equation,
   $$\max_X \sum_{k=1}^{l} \text{trace}\left(X^T M_k X\right) = \max\left[\text{trace}\left(X^T\left(\sum_{k=1}^{l} M_k\right)X\right)\right],$$

such that, $X^TX = I$. Once $\overline{M}^t$ is calculated, we can perform SVD on $\overline{M}^t$ and extract the d principal singular vectors as the embedding, X^t, for $\mathbf{G}^t$.

4. **WSUMU**

   Another way to combine the slices of $\mathbf{M}^t$ is to calculate a weighted sum of utility matrices (WSUMU),

$$\widetilde{M}^t = \sum_{k=1}^{l} w_k M_k^t,$$

   where $\mathbf{w} = [w_1, \ldots, w_l]$ denotes a l-dimensional vector of weights calculated from the data, and $\|\mathbf{w}\|_F = 1$. Adapting the ideas of Liu et al. (2013, section 3.2.3.), an embedding, X, from $\widetilde{M}$ is obtained by solving

$$\max_{X,\mathbf{w}} \sum_{k=1}^{l} w_k \text{trace}(X^T M_k X),$$

   which is equivalent to

$$\max_{X,\mathbf{w}} \text{trace}\left(X^T \left(\sum_{k=1}^{l} w_k M_k\right) X\right),$$

   such that $X^TX = I$, $\mathbf{w} \geq 0$, and $\|\mathbf{w}\|_F = 1$. We employ the strategy given in Liu et al. (2013) (summarized in Algorithm 7) to construct $\widetilde{M}^t$ and obtain an embedding. The matrix at convergence is the embedding, X^t, of the multi-graph, $\mathbf{G}^t$.

5. **AVGF**

   Tang et al. (2012) employ CCA (canonical correlation analysis) to calculate an average from a set of embeddings after adjusting for Euclidean transformations. Let $X_k \in \mathbb{R}^{n \times d_k}$ denote the embedding extracted from G_k, and $R_k \in \mathbb{R}^{d_k \times d}$ be a linear transformation applied to X_k. Tang et al. (2012) propose to combine the embeddings as

$$\bar{X} = \frac{1}{l} \sum_{k=1}^{l} X_k R_k. \tag{4.16}$$

   By assuming that the embeddings are highly correlated after the application of linear transformations, CCA attempts to find $R_1, \ldots, R_l$ that maximize the summation of correlations between each pair of embeddings. The CCA objective function is defined as

$$\max \sum_{k=1}^{l} \sum_{k'=1}^{l} {R_k}^T C_{kk'} R_{k'}, \tag{4.17}$$

Algorithm 7 Spectral Embedding using WSUMU Method

Input: (i) Tensor, $\mathbf{W}$, of dimensions $n \times n \times l$, where the mode-3 slices are the weighted adjacency matrices, $W_1, W_2, \ldots W_l$, (ii) threshold, ϵ

Output: Final embedding, $X \in \mathbb{R}^{n \times d}$

1: **for** $k = 1$ **to** l **do**
2: Update: $W_k = \log_{10}(W_k + \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)$
3: Update: $W_k = \frac{W_k}{\max_{i,j}\{W_{i,j,k}\}}$
4: Calculate $\tau_k = \frac{1}{4n^2}\sum_{i,j} W_{i,j,k}$
Update: $W_{(k,\tau)} = W_k + \tau_k \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T$
Calculate $D_{(k,\tau)}$ with diagonal elements, $[D_{(k,\tau)}]_{i,i} = \sum_{j=1}^{n}[W_{(k,\tau)}]_{i,j}$
Calculate $M_k = D_{(k,\tau)}^{-1/2} W_{(k,\tau)} D_{(k,\tau)}^{-1/2}$
5: **end for**
6: Calculate $\tilde{M} = \sum_{k=1}^{l} M_k$.
7: Perform SVD, $\tilde{M} = U\Sigma U^T$
8: Estimate d by applying Algorithm 1 on $\tilde{M}$
9: Initialize $U_0 = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$
10: Initialize $\mathcal{D} = \inf$
11: **while** $\mathcal{D} > \epsilon$ **do**
12: Calculate $\hat{\mathbf{w}}$: $\hat{\mathbf{w}} = [\text{trace}(U_0^T M_1 U_0) \ldots \text{trace}(U_0^T M_l U_0)]$
13: Calculate $\mathbf{w}$: $\mathbf{w} = \frac{\hat{\mathbf{w}}}{\|\hat{\mathbf{w}}\|_F}$
14: Calculate $\widetilde{M} = \sum_{k=1}^{l} w_k M_k$
15: Perform SVD, $\widetilde{M} = V\Sigma V^T$
16: Obtain $\hat{U} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$
17: Update $\mathcal{D} = \| U_0 - \hat{U} \|_F^2$
18: Update: $U_0 = \hat{U}$
19: **end while**
20: Obtain $X = U_0$

where each $C_{kk'}$ gives the covariance matrix corresponding to the pair of embeddings, X_k and $X_{k'}$, given by $X_k^T X_{k'}$. Since $X_k^T X_k = I$, expression 4.17 is subject to the constraint $\sum_{k=1}^{l} {R_k}^T C_{k,k} R_k = I$.

Tang et al. (2012) show that the average, $\bar{X}$, given in Equation 4.16, corresponds to the principal d left singular vectors of matrix, $Y = [X_1, \ldots, X_l]$. In Algorithm 8, we summarise our procedure of obtaining an average embedding, X, for a multi-graph using Tang et al. (2012)'s method.

Methods AVG, MAX, AVGU, WSUMU and AVGF all employ different strategies to convert a time sequence of tensors into a time sequence of embeddings. In each method, once a time sequence of embeddings is obtained, we can continue from step 4 of Algorithm 6 to calculate vertex change scores, $\mathbf{z}^t$.

It is equally important to know whether considering multiple views benefits the change detection procedure compared to using only a single view as described in Chapter 3. Let $W_k^1, \ldots, W_k^{\mathcal{T}}$ be the time sequence of weighted adjacency matrices obtained by extracting

Algorithm 8 Spectral Embedding using AVGF Method

Input: Tensor, $\mathbf{W}$, of dimensions $n \times n \times l$, where the mode-3 slices are the weighted adjacency matrices, $W_1, W_2, \ldots W_l$.
Output: Final embedding, $X \in \mathbb{R}^{n \times d}$

for $k = 1$ **to** l **do**
  Update: $W_k = \log_{10}(W_k + \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)$
  Update: $W_k = \frac{W_k}{\max_{i,j}\{W_{i,j,k}\}}$
  Calculate $\tau_k = \frac{1}{4n^2}\sum_{i,j} W_{i,j,k}$
  Update: $W_{(k,\tau)} = W_k + \tau_k \mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T$
  Calculate $D_{(k,\tau)}$ with diagonal elements, $[D_{(k,\tau)}]_{i,i} = \sum_{j=1}^{n}[W_{(k,\tau)}]_{i,j}$
  Calculate $M_k = D_{(k,\tau)}^{-1/2} W_{(k,\tau)} D_{(k,\tau)}^{-1/2}$
  Perform SVD: $M_k = U_k \Sigma_k U_k^T$
  Estimate d_k by applying Algorithm 1 on M_k
  Extract d_k principal singular vectors, X_k, from each U_k starting from the second vector onwards
end for
Construct $Y = [X_1, \ldots, X_l] \in \mathbb{R}^{n \times \sum_{k=1}^{l} d_k}$
Compute SVD of $Y = U\Sigma V^T$
Estimate d by applying Algorithm 1 on Y
Obtain $X = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$

slice–k from each $\mathbf{W}^t$. This sequence defines a single-view dynamic network with respect to view–k. By using $W_k^1, \ldots, W_k^{\mathcal{T}}$ as input to CDP (Algorithm 3 in Section 3.2.6), we calculate vertex change scores, $\mathbf{z}_k^t$, and calculate the performance measures, $\boldsymbol{\eta}_k^t$ (Equation 3.18) and $\bar{\boldsymbol{\eta}}_k^t$ (Equation 3.19). We compare the performance of MCDP–I and MCDP-II with CDP applied to each view to investigate how multiple views affect performance of change detection.

4.2.2 Results

For each change scenario discussed in Section 4.2, we first calculate the performance measure $\boldsymbol{\eta}^{11}$ (Equation 3.18) using the change scores resulting from applying CDP separately to each slice of the tensor, and applying MCDP-I, MCDP-II, AVG, MAX, AVGU, WSUMU, and AVGF to the whole tensor. If $\boldsymbol{\eta}^{11}$ is positive, then this indicates that the change scores for vertices in V_c are higher than the change scores for the rest of the vertices in $V_{\bar{c}}$ at the time instant of change, $t = 11$ (Section 3.3.4). For each change scenario, we then calculate the performance measure $\bar{\boldsymbol{\eta}}^t$ (Equation 3.19) for several time instants before and after $t = 11$ for all multi-graph methods.

Figure 4.3 shows $\boldsymbol{\eta}^{11}$ for group-change in the simple situation. We use CDP-1, CDP-2, CDP-3 and CDP-4 to indicate the performance of change detection using CDP with respect to view-1, view-2, view-3, view-4, respectively. It can be seen that the first

six multi-view methods clearly outperform those on single views. Methods AVGF and MCDP-II both combine the embeddings separately obtained from each slice of the tensor by employing CCA and GPA, respectively. However, the bulk of the $\boldsymbol{\eta}^{11}$ interval of MCDP-II lies above 1.5, while $\boldsymbol{\eta}^{11}$ of AVGF lies below zero. This shows that MCDP-II substantially outperforms AVGF in detecting the change. Hence, GPA is better than CCA for obtaining an average from a set of embeddings. The $\bar{\boldsymbol{\eta}}^{t}$ intervals in Figure 4.4 show a clear detection at the changed time instant for all methods except AVGF. From the above results, considering multiple views benefits change detection performance in the simple situation. From Figure D.1, $\mathbf{W}$ generated in the simple situation has the same block structure shared across all slices. Utilizing information from all slices helps to compensate for shortfalls in information with respect to single slices, and obtain a more reliable embedding of the graph that in turn improves change detection performance.

We observe that AVG and MAX perform best in the group-change scenario in the simple situation. These methods combine the slices directly without performing pre-processing or normalization as do the other methods. Since the simple situation generates slices such that the same block structure is shared across all, the resulting matrix $\overline{W}$ or $W_{\max}$ (Section 4.2.1) gives a clearer representation of the inlying relationship structure. Hence, the improvement in change detection performance. However, as slice-wise normalization of entries is not applied to AVG and MAX, these two methods are not comparable to the others. For example, let us show the sensitiveness of AVG and MAX to the scale of the elements in the slices. From Figure 4.3, slice–3 (CDP-3) has the worst performance with respect to $\boldsymbol{\eta}^{11}$. Let us redo the same experiment (group-change scenario in simple situation) by multiplying the elements of slice–3 by 100. Figure 4.5 compares the $\boldsymbol{\eta}^{11}$ intervals of the seven multi-view methods before and after the multiplication. Clearly, there is a considerable decrease in performance of AVG and MAX after emphasizing slice–3, while other methods hardly show any difference. To nullify the effect of scaling in AVG and MAX, a natural solution is to normalize each slice by $\frac{W_k}{\|W_k\|_F}$, prior to calculating $\overline{W}$ or $W_{\max}$. From this point onwards, we apply this normalization step prior to combining the weighted adjacency matrices for AVG and MAX methods.

We next investigate change detection performance in the presence of noisy slices (the noisy situation) and highly variable block structure among slices (the complex situation). Figure 4.6 compares $\boldsymbol{\eta}^{11}$ calculated for the group-change scenario in all simple, noisy and complex situations. From Figure 4.6 (top), CDP–1 shows similar performance in all situations. This is expected as slice–1 has a clear three block structure in all situations (Figures D.1, D.5 and D.9). From 4.6 (bottom), except for AVGF, all multi-view methods show good detection performance in all situations. In the simple situation, our proposed multi-view methods and other comparison methods, excluding AVGF, outperform all single-view methods. We observe how slice–3 alone (CDP–3) shows failure in

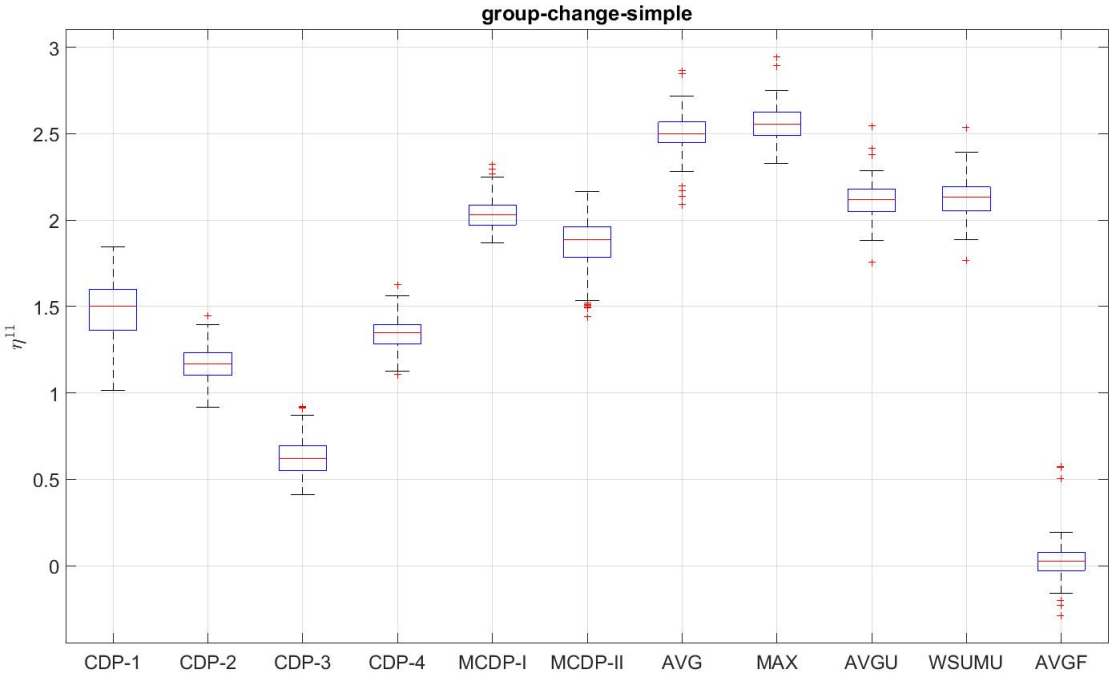


FIGURE 4.3: **Observing $\boldsymbol{\eta}^{11}$ for group-change in simple situation for $w = 5$.** Except for AVGF, all of the multi-view methods outperform the single-view methods.

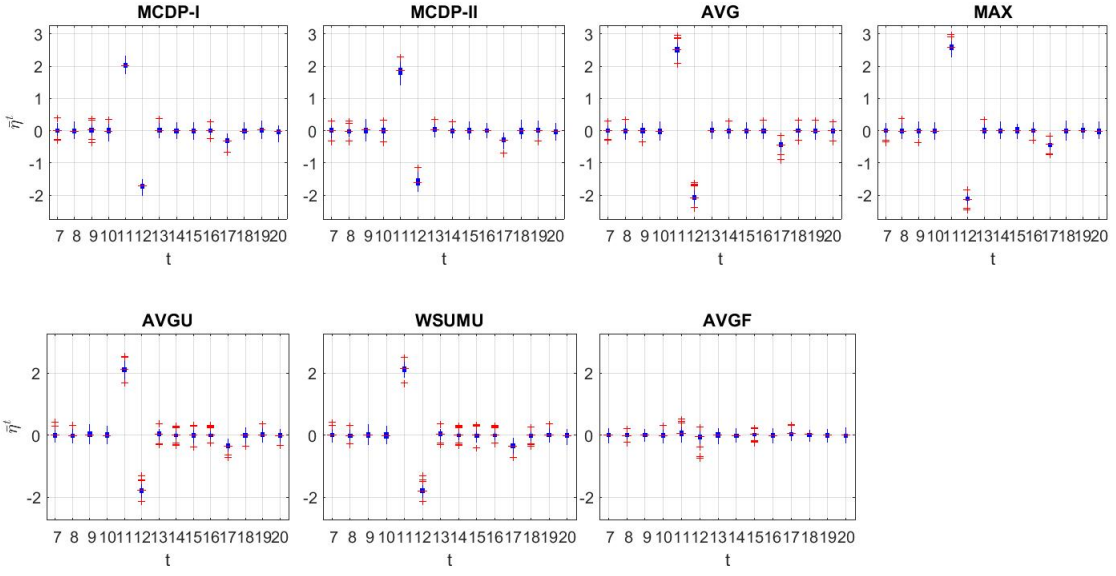


FIGURE 4.4: **Observing $\bar{\boldsymbol{\eta}}^t$ for group-change in simple situation for $w = 5$ change point.** Except for AVGF, all methods show a clear detection at $t = 11$.

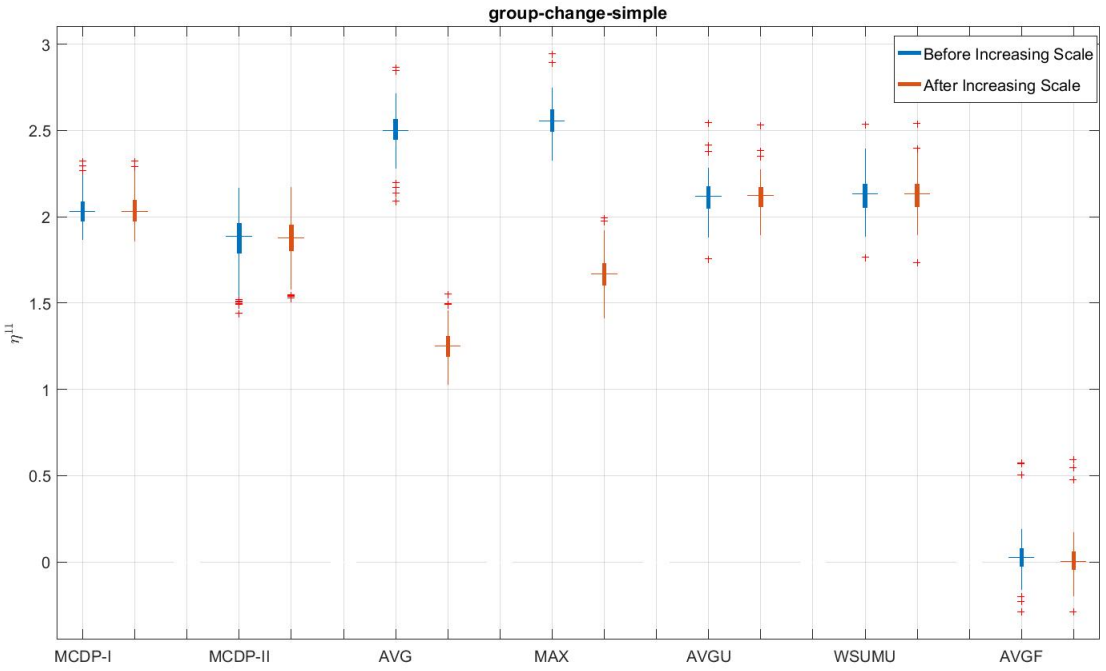


FIGURE 4.5: **Observing effect of the scale of slices towards performance.** Only AVG and MAX methods show decrease in performance after emphasizing the slice with least performance.

detection in the complex situation, but MCDP–I and MCDP–II correct this problem by using the information in other slices. For the group-change scenario, MCDP-I outperforms MCDP-II in all simple, noisy and complex situations. MCDP-I and MCDP-II do not perform better than AVG, MAX, AVGU and WSUMU in this experiment. MAX shows best performance in simple and complex situations. In the noisy situation, AVG and MAX show similar performance. We do not see a difference in the performances of WSUMU and AVGU in the simple situation. However, in noisy and complex situations, WSUMU shows better performance than AVGU. The justification for this is that WSUMU penalizes slices with unclear block structure. We refer the reader to Appendix D for sign test results as statistical evidence for the above conclusions.

We further calculate $\bar{\boldsymbol{\eta}}^t$ for several time instants before and after $t = 11$ for the group-change scenario in simple (Figure D.14), noisy (Figure D.15) and complex (Figure D.16) situations. All multi-view methods except AVGF show a clear detection in all three situations.

Figure 4.7 shows $\boldsymbol{\eta}^{11}$ for the fragment scenario. The bulk of $\boldsymbol{\eta}^{11}$ of AVGF in all situations lies below zero. Thus, AVGF fails to detect the fragment scenario in all situations. We observe that MCDP-I and MCDP-II effectively exploit the information from all slices to detect the changed vertices. For example, eventhough CDP–2 in the complex situation

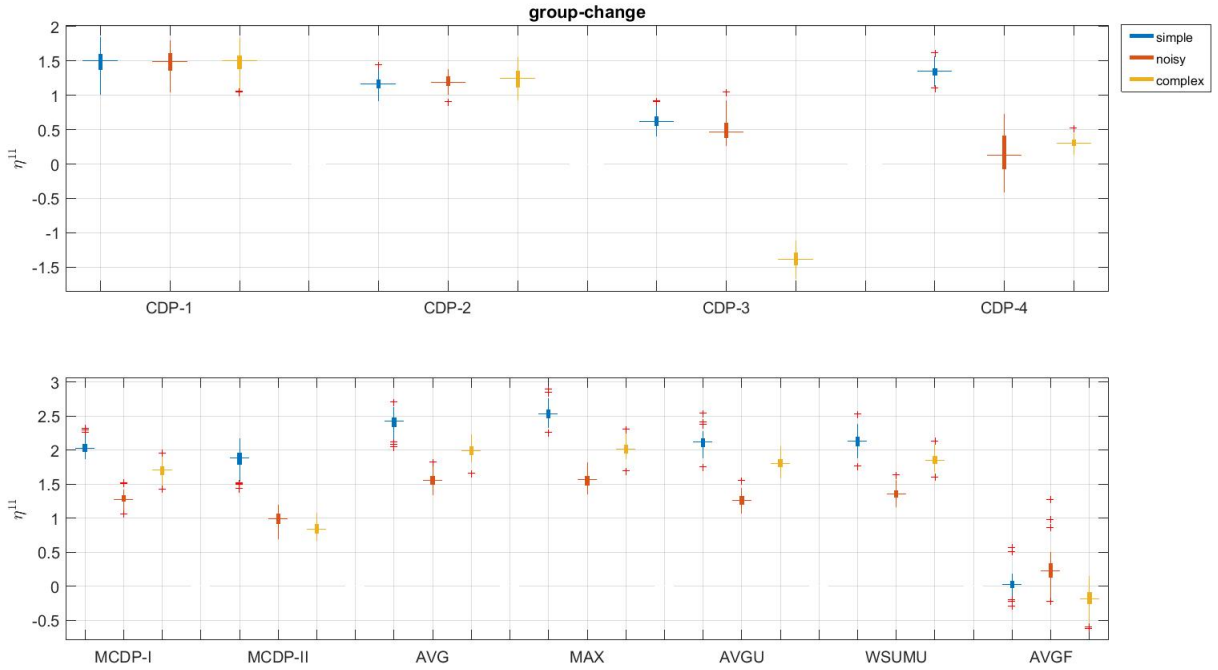


FIGURE 4.6: Comparison of $\boldsymbol{\eta}^{11}$ intervals for group-change in the simple, noisy, and complex situations for $w = 5$.

and CDP–3 and CDP–4 in the noisy situation show failure in detection, MCDP-I and MCDP-II show good change detection performance. MCDP-I outperforms MCDP-II, AVGU, and WSUMU in all situations, and the AVG and MAX methods in the noisy and complex situations. MCDP-II outperforms AVGU and WSUMU in all situations, AVG in the noisy situation and performs equal to MAX in the noisy situation.

Figure 4.8 shows $\boldsymbol{\eta}^{11}$ for the hetero-to-homo scenario. Similar to the group-change and fragment scenarios, MCDP-I and MCDP-II use the information from good slices to adjust for weak slices resulting in good change detection performance. MCDP-I outperforms all methods except AVG in simple situation, all methods in noisy situation and all methods except MCDP-II and AVG in complex situation. MCDP-I performs equal to AVG in complex situations. MCDP-II outperforms all methods for the hetero-to-homo scenario in the complex situation.

In order to confirm a detection at $t = 11$, we then observe $\bar{\boldsymbol{\eta}}^t$ for several time instants before and after the change point, $t = 11$, for the fragment (Figures D.17, D.18, D.19) and hetero-to-homo (Figures D.20, D.21, D.22) scenarios. The results show that MCDP-I, MCDP-II, AVG, MAX, AVGU and WSUMU clearly detect both the fragment and hetero-to-homo scenarios in all situations. AVGF on the other hand, shows failure in

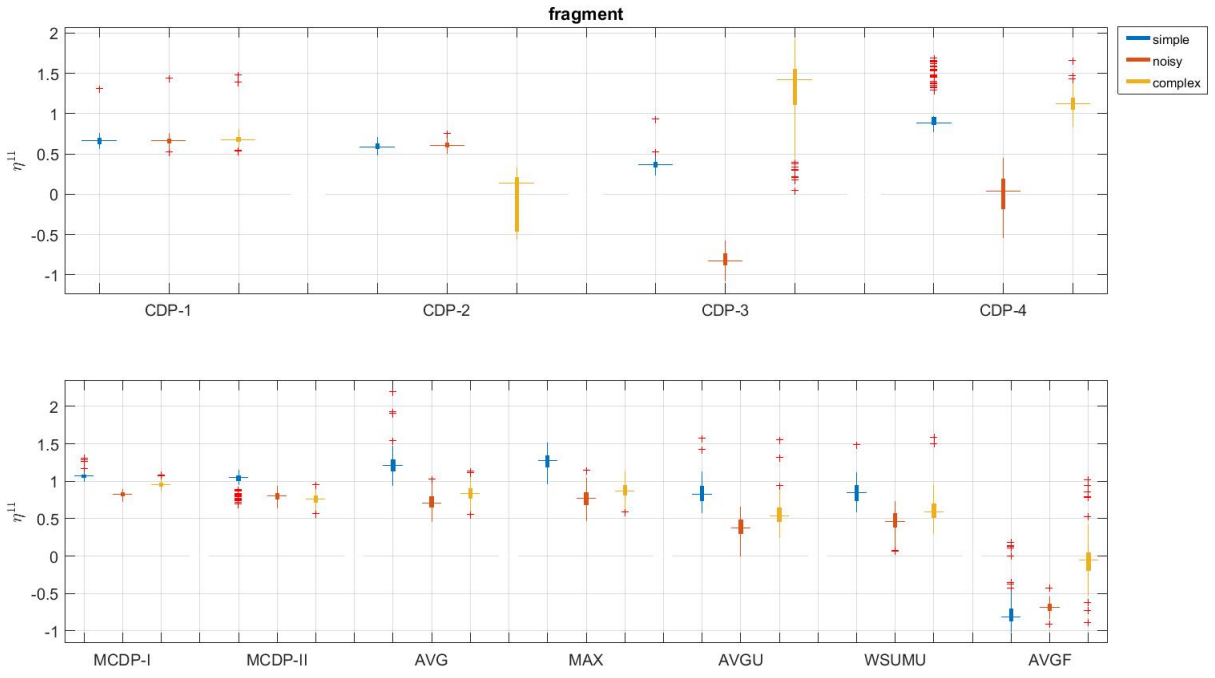


FIGURE 4.7: Comparison of $\boldsymbol{\eta}^{11}$ intervals for fragment in the simple, noisy and complex situations for $w = 5$.

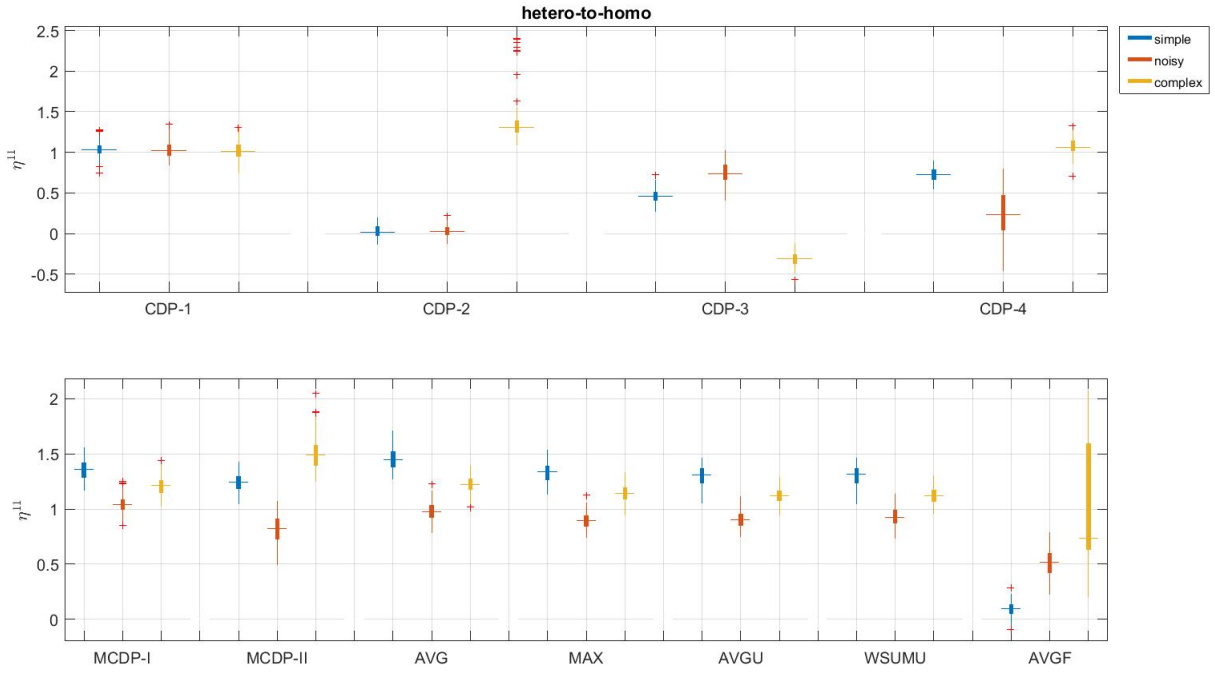


FIGURE 4.8: Comparison of $\boldsymbol{\eta}^{11}$ intervals for hetero-to-homo in the simple, noisy and complex situations for $w = 5$.

detection in all three situations for the fragment scenario. However, for the hetero-to-homo scenario in all situations, we observe a slight increase in $\bar{\boldsymbol{\eta}}^{11}$ compared to $\bar{\boldsymbol{\eta}}^{10}$ for AVGF.

Next, we investigate the most difficult case, the mixed scenario, where vertices in different slices undergo different changes in the complex situation. We calculate vertex change scores for mixed-scenario using all multi-view methods. We compare performance of detecting vertices, $1, \ldots, 300$ (*All*), that change in all slices, and detecting vertices $301, \ldots, 600$ (*Two*), that change only in two of the four slices. Figure 4.9 shows $\boldsymbol{\eta}^{11}$ corresponding to this mixed-scenario. From the results, MCDP-I and MCDP-II show highest performance in detecting both types of vertices *All* as well as *Two*, while MCDP-II is the best. We further calculate the $\bar{\boldsymbol{\eta}}^{t}$ for several time instants before and

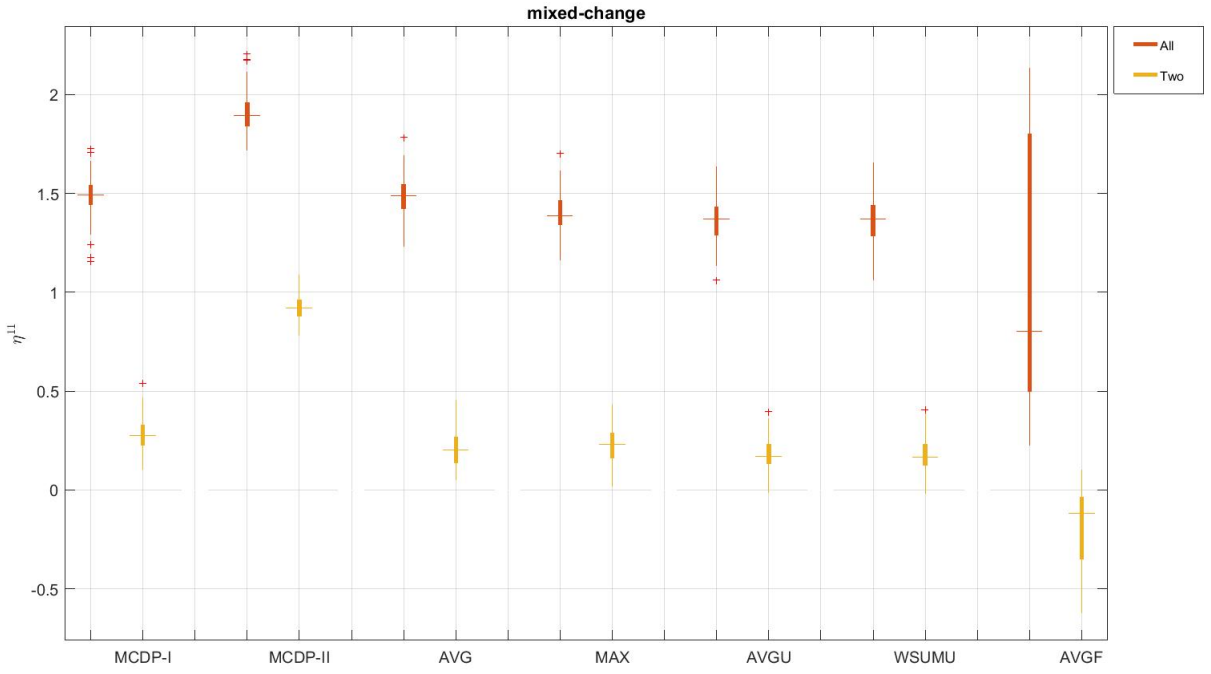


FIGURE 4.9: Comparison of $\boldsymbol{\eta}^{11}$ intervals for mixed-scenario in complex situation for $w = 5$.

after $t = 11$, for *All* (Figure D.23) and *Two* (Figure D.24). MCDP-II shows good performance in detecting vertices in both cases. MCDP-I shows good detection performance in detecting vertices in *All*, but not in *Two*. AVG, MAX, AVGU and WSUMU only detect vertices in *All*. AVGF also shows an increase in $\bar{\boldsymbol{\eta}}^{11}$ in both cases. However, as we observe $\boldsymbol{\eta}^{11}$ to be mostly negative for case *Two* in Figure 4.9, AVGF fails to detect vertices that change only in two slices.

Next we perform experiments to evaluate the scalability of our proposed multi-view methods compared to the baseline methods. Recall that the difference between these

methods lies in the way an embedding is obtained from the tensor. Thus we evaluate scalability by comparing the computational times of embedding a tensor in each method. For this, tensors representing multi-graphs of sizes, $300, 900, 1500, 2100, 2700$ and 3300, are generated and the time it takes for each method to obtain an embedding is calculated. This is repeated for 100 simulation runs and the resulting averages are calculated. We do not evaluate the scalability of AVGF as it failed to detect changes in most of our simulation experiments shown previously in this section. Our final results are given in Table 4.4 and are further illustrated in Figure 4.10. All experiments are implemented on a Windows server Intel Xeon with two 3.3GHz processors of 128 GB RAM.

TABLE 4.4: Average CPU time taken by each multi-view method

n	MCDP-I	MCDP-II	AVG	MAX	AVGU	WSUMU
300	1.7478	1.9394	0.7044	0.5448	0.6208	0.8895
900	15.0848	8.4393	2.2527	1.9844	2.6649	2.9320
1500	43.0166	16.6829	4.5397	4.9692	4.6616	6.9305
2100	77.0079	44.2871	10.7968	10.4375	12.4408	17.3457
2700	125.8545	85.2853	19.4803	19.8824	26.7110	34.0152
3300	263.8908	235.0326	53.8989	54.4292	67.5710	90.4192

Clearly, AVG, MAX, and AVGU that directly average the slices of the weighted adjacency tensor or the representation tensor are the most computationally efficient. WSUMU takes comparatively more time than AVGU, as it calculates weights before combining the slices. MCDP-II requires the calculation of an embedding from each slice, hence the higher computational cost. However, this drawback can be addressed by using parallel computing. MCDP-I, which performs tensor decomposition, takes the highest computational time for all graph sizes.

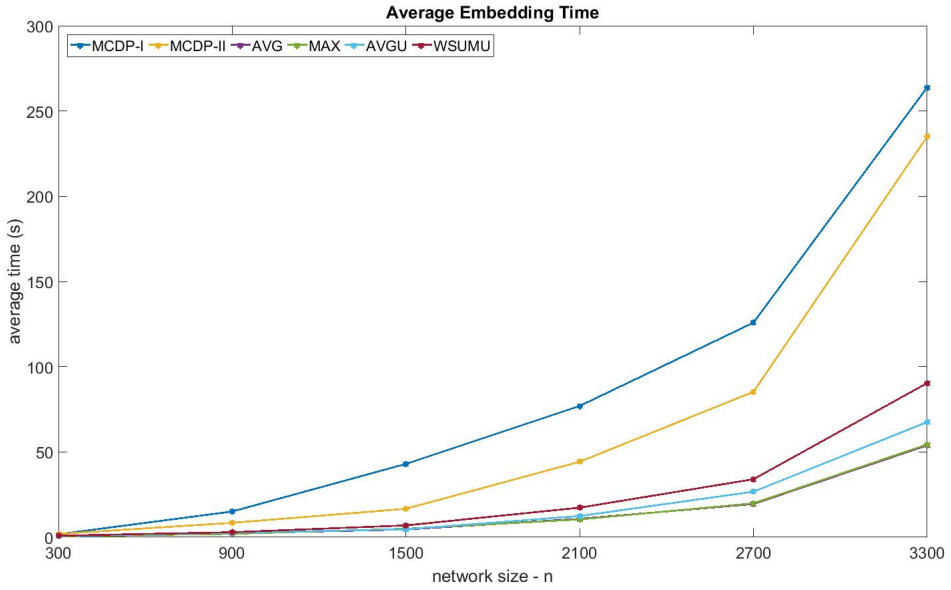


FIGURE 4.10: Comparison of average embedding times of different multi-view methods.

From all of the simulation results, MCDP-II shows the best change detection performance when the connectivity structure varies across different slices of the tensor and vertices undergo different changes in each slice. MCDP–I generally performs better otherwise.

4.3 Case Study: The Enron E-mail Network

In Chapter 3, we evaluated the performance of CDP (Algorithm 3) by applying it to the Enron email network (Section 3.4). We used a cleaned and processed version of the original Enron dataset from Tang et al. (2008) for this purpose. This dataset contains an email-sender dynamic network as well as an email-receiver dynamic network over 2359 user email addresses. As CDP could only handle graphs represented as weighted adjacency matrices, we suppressed the attributes on the connections which indicated the direction of the emails (sent or received), and extracted a time sequence of weighted adjacency matrices. In this section, we further take into account the edge attributes, construct a time sequence of tensors, and perform change detection using our proposed multi-view methods.

Using the available information, we first construct a time sequence of directed graphs encoding sender-receiver relationships. Using the directed graph at each time instant, we then extract two undirected graphs to separately capture the sending and receiving patterns of the 2359 users. In one graph, we connect two vertices if the corresponding

users have a recipient in common. In the other graph, we connect two vertices if the corresponding users receive emails from the same senders. This way, we obtain a time sequence of undirected multi-graphs, where each time instant corresponds to one month, and the whole time period spans December 1999 to March 2002. The vertices in each multi-graph represent the 2359 users. The edge weights in the graph corresponding to sending patterns denote the number of times two users have a common receiver and the edge weights in the graph corresponding to receiving patterns denote the number of times two users have a common sender respectively. We denote the time sequence of multi-graphs as a time sequence of three-dimensional weighted adjacency tensors. At each time instant, slice–1 of the tensor corresponds to the undirected graph (graph–1) denoting sending patterns, while slice–2 corresponds to the undirected graph (graph–2) denoting receiving patterns.

As in Section 3.4, we evaluate the performance of our multi-view change detection methods based on the hypothesis that email communication patterns within the company are affected by the events associated with the scandal. We apply MCDP-I and MCDP-II on the time sequence of tensors with a window, w, of length 1, and calculate vertex change scores at each time instant. Similar to our analysis in Section 3.4, at each time instant, we convert the change scores to z-scores and threshold z-scores to detect vertices who change most at a given time instant. Thorough investigation shows a threshold of 3.5 to be suitable to keep the percentage of vertices detected at each time instant to be below two percent.

Kenneth Lay, the founder of Enron, is ranked twice as one of the most changed entities by MCDP-II. Lay is first detected during the transition from April 2001 to May 2001. According to the Enron time-line[2], Enron scheduled an unusual analyst conference call to boost stock during this transition. Figure 4.11 shows the degree of the vertex corresponding to Lay in graph–1 (top) and graph–2 (bottom), over the whole time period. We observe a considerable increase in the degree in May-2001 in graph–2. Figure 4.12 also shows the subgraph for the vertex corresponding to Lay during the same transition in graph–1. We observe that Lay is connected to top level executives (who were also involved in the scandal) such as Andrew Fastow, Kevin Hannon, Rod Hayslett and John Lavorato during April-2001. This shows that Lay's email sending patterns were similar to many top level executives at that time instant. In May-2001, we observe that Lay's connections are mostly to employees with different job roles in the company. This provides further evidence for the detected change in Lay's email sending patterns during that transition.

[2] http://www.agsm.edu.au/bobm/teaching/BE/Enron/timeline.html

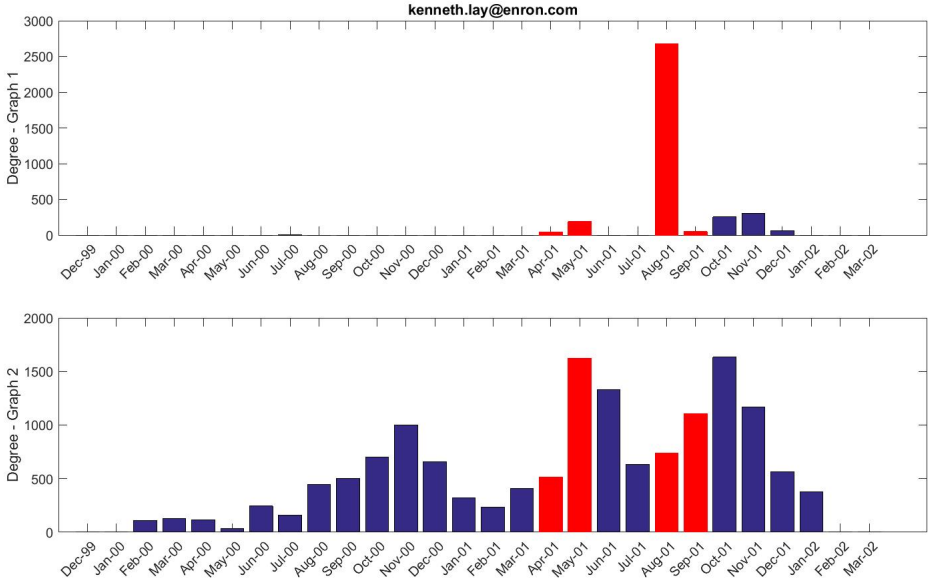


FIGURE 4.11: The degree of the vertex representing Kenneth Lay with respect to graph–1 and graph–2 during the 28 months

Lay's second detection corresponds to the transition from August-2001 to September-2001. August-2001 is the time when Enron's accounting practices were questioned and Lay was reappointed as CEO. In September-2001, Enron stock sales regained good position and activities were more normal. We observe in Figure 4.11 (top), the high degree of the vertex in August-2001, which then drops in September-2001. MCDP-I also detects Lay among the vertices which have changed most for the same period. Another change detected by MCDP-II is Mark Koenig (former head of investor relations and a major player in the Enron scandal). Koenig is detected among the mostly changed entities during the transition from May-2000 to June-2000. This coincides with the time when the Enron broadband unit joined forces with Blockbuster, to supply video-on-demand. Figure 4.13 shows the variation in the degree of the vertex corresponding to Koenig, in both graph–1 (top) and graph–2 (bottom), throughout the whole time period. We do not observe a significant change in degree during May-2000 to June-2000. Figure 4.14 shows the subgraph centred at the vertex corresponding to Koenig during the transition in graph–2. We observe how Koenig mainly interacts with the top level executives (such as, Jeff Skilling, Richard Causey, Paula Rieker, also major players in the scandal) in the time instant corresponding to May-2000. In June-2000, Koenig interacts with many people with different job roles in the company. MCDP-II also detects Kenneth Rice (former broadband unit CEO) during the same transition period. Mark Koenig is again detected by MCDP-II at time instant corresponding to February-2002. From Figure 4.13, we observe how Koenig ceases all his interactions in both graph–1 and graph–2 during this transition.

FIGURE 4.12: **Subgraph corresponding to Kenneth Lay during April-May 2001**. Lay is represented by the enlarged blue vertex in the centre and the edges connected to him are highlighted in red.

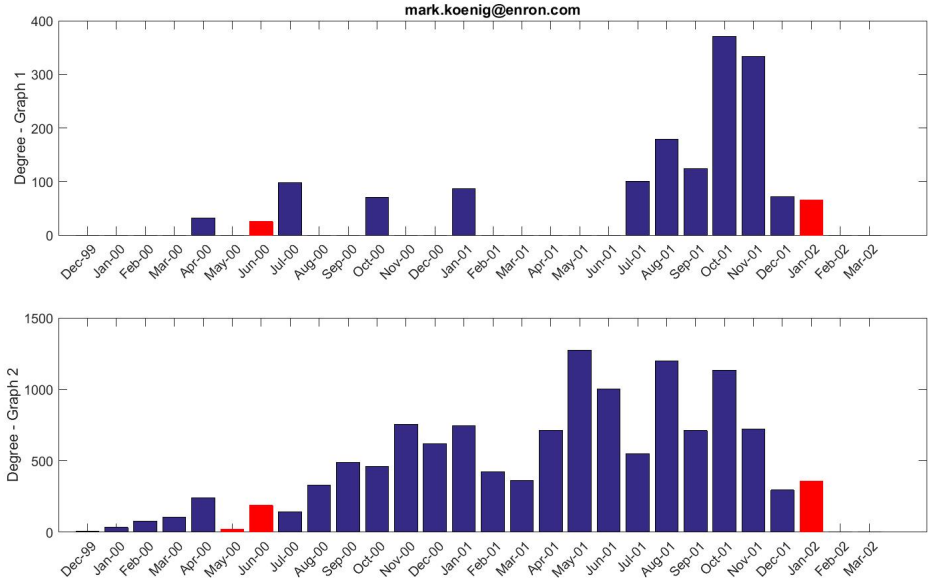


FIGURE 4.13: The degree of the vertex corresponding to Mark Koenig with respect to graph–1 and graph–2 during the 28 months

MCDP-II detects Ben Glisan (former Enron treasurer), during the transition from August-2000 to September-2000. This corresponds to the time when Enron stock hits an all-time high and the Federal Energy Regulatory Commission orders an investigation into Enron's strategies. Figure 4.15 shows the degree of the vertex corresponding to Ben Glisan over the whole time period with respect to both graph–1 and graph–2. We observe a decrease in the number of edges connected to that vertex in graph–1, but an increase in the number of edges in graph–2 at the detected time instant. MCDP-II successfully detects the change eventhough the vertex undergoes different changes in the two graphs.

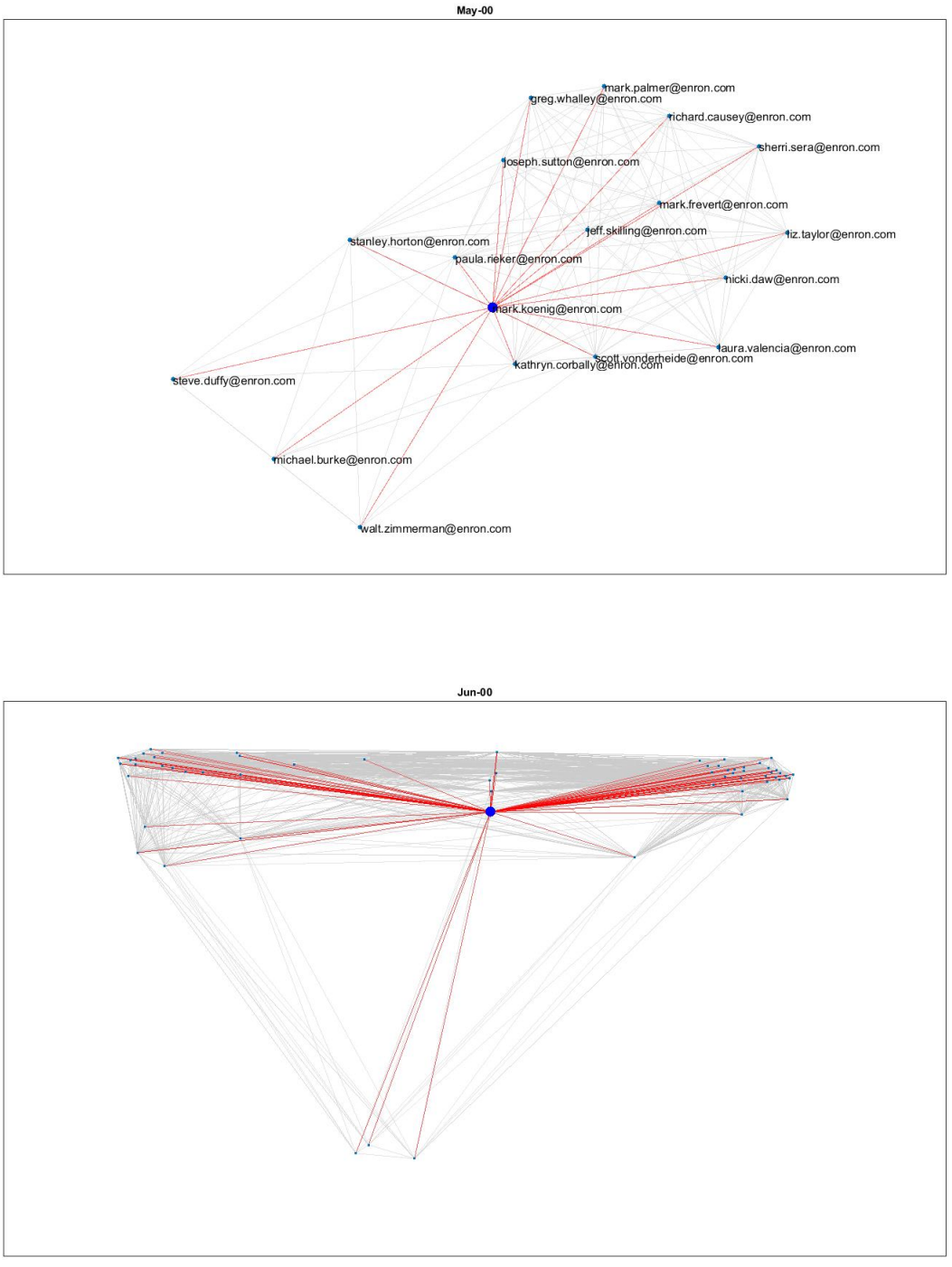


FIGURE 4.14: **Subgraph of Mark Koenig during May-June 2000**. Koenig is represented by the enlarged blue vertex in the centre and the edges connected to it are highlighted in red.

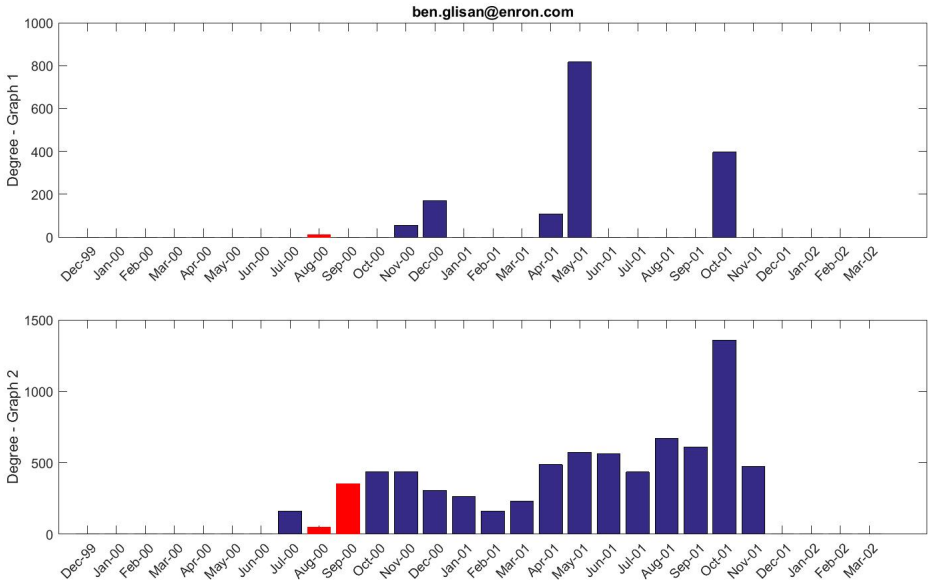


FIGURE 4.15: The degree of the vertex corresponding to Ben Glisan with respect to graph–1 and graph–2 during the 28 months

MCDP-II further detects Christopher Calger (executive in Enron's trading business), David Delainey (head of Enron's trading and energy units) and Timothy Beldon (former top Enron trader) during February-2000 to March-2000. During this time period, Enron declared its Broadband plan which in turn attracted more investors. Thus, the change in the behaviour of the three fraudsters who were mainly involved in trading is justifiable.

Based on the analysis of the Enron email network, we find MCDP-II as the overall winner based on detection performance. MCDP-II successfully detected several fraudsters involved in the Enron scandal at meaningful time instants, while MCDP-I was able to detect only one of them.

4.4 Summary

In this chapter, we extend the CDP framework proposed in Chapter 3 to handle multiple edge attributes represented as tensors. Given a time sequence of multi-graphs, each represented as a tensor, our main goal is to detect vertices that change behaviour with respect to one or more edge attributes at any time instant. Our main contributions are the two proposed methods to obtain an embedding from a tensor. The first method, MCDP-I, employs the HOSVD method to factorise the tensor and obtain a single embedding directly. The second method, MCDP-II, on the other hand, obtains separate embeddings from each slice of the tensor and then employs GPA techniques to average them to produce a single embedding. We assessed the performance of our proposed

methods by conducting extensive simulation experiments and applying the methods to a real-world dataset. We compared our methods with five other methods that differ from each other in the way that a single embedding is produced from different slices of the tensor.

Both MCDP-I and MCDP-II clearly showed good performance in detecting changed vertices in all experiments conducted. However, in a tensor representing a multi-graph with complex structure, MCDP-II showed better performance than MCDP-I, especially in detecting vertices that do not exhibit change consistently across all slices. Baseline methods such as AVG, MAX, AVGU and WSUMU disregard the multi-graph aspect when obtaining an embedding. Those methods perform poorly in difficult situations, where vertices change in different ways, and the graphs corresponding to each slice contain substantially different connectivity structure. Another interesting finding from our analysis is the superiority of the GPA technique over the CCA technique for obtaining an average from a set of embeddings. AVGF proposed in Tang et al. (2012), which is based on the CCA principle, did not show good detection performance in most of the experiments.

We also investigated whether combining the information from multiple slices is beneficial compared to performing change detection separately on single slices. We saw that MCDP-I and MCDP-II generally outperformed CDP on single slices in the simple situation, where all the slices of the tensor depict a similar connectivity structure. However, in noisy and complex situations, MCDP-I and MCDP-II sometimes showed poorer performance than CDP on some slices. This seems to suggest that combining slices that contain conflicting information can degrade change detection performance in multi-view methods. Thus, an interesting future research problem would be to find a way to determine which slices to use and which to omit. Furthermore, for MCDP-I we use HOSVD for decomposing the tensor, but it is not the only tensor factorization method that can be used. It would be interesting to explore whether using other tensor decomposition methods would be more advantageous for change detection. A comprehensive review of different tensor decomposition methods can be found in Kolda and Bader (2009).

To conclude, in this chapter, we presented two tensor-based embedding methods, MCDP-I and MCDP-II, for change detection in multi-view networks. Both methods successfully detect a wide range of vertex-based changes that closely relate to changes that can occur in real-world dynamic multi-view networks.

Chapter 5

Concluding Remarks

Recently there has been a burgeoning interest in network-based methods for analysing large and complex datasets. In this thesis we have demonstrated methods for change detection by representing entities and their time-varying relationships as a time sequence of graphs. Our general approach has been developed with two main objectives:

(i) detecting changes involving vertices in a time sequence of graphs based on matrix-based spectral embedding methods, and
(ii) extending (i) for multi-graphs represented as three-dimensional tensors.

A comprehensive description of the theoretical background necessary to achieve these objectives were provided in Chapter 2. In Chapter 3 and Chapter 4, we proposed methodologies to accomplish our objectives, and explored their performance in a range of simulated situations as well as in a real life case study. In this chapter, we provide a summary of the overall work of this thesis and explain its practical implications. We also discuss future research directions suggested by the results obtained from our experiments.

5.1 Overall Summary

We have developed a novel method called CDP to detect changes in a dynamic network that is represented as a time sequence of weighted graphs. We employed spectral embedding methods to study the behaviour of vertices in these graphs. A majority of existing spectral-based change detection methods, such as Idé and Kashima (2004), do not account for sparseness or degree heterogeneity when obtaining an embedding for the graph. Because of this, when applied to sparse and heterogeneous graphs, the resulting

embedded points only focus on high degree vertices, providing misleading change detection results. We adapted techniques in spectral graph theory to formulate an embedding procedure that can handle sparse and heterogeneous graphs. Our characterization of a change in a vertex's behaviour centres around the variation of its position in the latent Euclidean space over a series of time instants. In order to quantify changes we incorporated Procrustes analysis techniques. To the best of our knowledge, the use of Procrustes analysis for the purpose of change detection in dynamic networks is new and was inspired by Tang et al. (2014) who use it to compare two random dot product graphs.

We demonstrated the performance of CDP using nine simulation studies and one real-world dataset, with comparison to two baseline methods, ACT (Idé and Kashima, 2004) and ACTM, which is a slight modification to ACT. CDP successfully detected various types of changes including (i) changes in vertex degree, (ii) changes in community membership, and (iii) unusual increase or decrease in edge weight between vertices. The ACT and ACTM methods only detected one scenario involving a change in the behaviour of a densely connected subgraph. From the overall simulation results, while ACTM showed slightly better performance than ACT, CDP was the best overall. We also applied CDP, ACT and ACTM to an undirected email network, derived from a subset of the Enron email dataset, which represented the monthly email communications between employees of the company. CDP successfully detected several top players in the scandal, who showed a large amount of change in their behaviour at time instants that coincided with key events in the Enron time-line. As most of those top players were company executives and traders who generally had very low email activity, changes in their behaviour were disregarded by ACT and ACTM.

In many situations, multi-view networks over the same set of entities are available. When using CDP to detect changes in a time sequence of multi-graphs, it is necessary to somehow aggregate the information on multiple edge attributes into a single weighted adjacency matrix before feeding the inputs into the algorithm. For example, when analysing the Enron email network using CDP, we had to suppress the information on the direction of the connections and obtain a symmetric weighted adjacency matrix at each time instant. However, such an approach may not be ideal when we want to maximally exploit the available information. We can address this problem by representing the connectivity structure of a multi-graph as a three-dimensional tensor by stacking the weighted adjacency matrices from each graph along the third dimension. The second part of this thesis extended CDP to handle tensors.

We represented a multi-view dynamic network as a time sequence of tensors. We proposed two strategies to obtain a low dimensional embedding from a tensor for use in

vertex-based change detection. Motivated by Liu et al. (2013)'s use of tensor factorization for spectral clustering on multi-view networks, our first strategy, MCDP-I, employs HOSVD to factorize the tensor at each time instant. Our second strategy, MCDP-II, first embeds each slice of the tensor separately using matrix factorization and then applies GPA techniques on the resulting set of embeddings to obtain an average embedding. Tang et al. (2012) employ a similar embedding averaging procedure to detect clusters in a multi-view network; however, they use CCA to obtain an average. In both MCDP-I and MCDP-II, after a time sequence of embeddings is obtained, the same Procrustes analysis procedure employed in CDP is used to obtain profile embeddings and thereafter to calculate change scores for vertices.

We have examined several situations to evaluate change detection performance of MCDP-I and MCDP-II. During our simulation experiments, we generated a time sequence of three-dimensional tensors in three situations: (i) simple, (ii) noisy and (iii) complex. These situations range from easy to difficult for the purpose of detecting vertices that have under-gone change. In all experiments, our proposed methods were compared against several other methods that differed from each other mainly in the way that information from different slices of the tensor were combined to obtain a single embedding. These included methods that combined (i) the weighted adjacency matrices (AVG (Han et al., 2015), MAX), (ii) the representation matrices (AVGU (Tang et al., 2009), WSUMU (Liu et al., 2013)), and (iii) the embeddings (AVGF (Tang et al., 2012)). Additionally, we also investigated whether it is advantageous to exploit multiple views rather than single views for change detection. Both MCDP-I and MCDP-II successfully detected all types of vertex-based changes that were considered in the experiments. When the connectivity structure varied considerably across the slices, MCDP-I and MCDP-II showed better performance compared to others. AVG, MAX, AVGU, WSUMU and AVGF generally failed to detect vertices that had changed in different ways in different slices. It was interesting to observe that although MCDP-II and AVGF both combine the single-slice embeddings, AVGF failed to detect the changes in most of the experiments. This shows the superiority of our proposed GPA technique against Tang et al. (2012)'s CCA technique for obtaining an average from a set of embeddings. We also applied MCDP-I and MCDP-II on the multi-view Enron email network that captured both incoming and outgoing email patterns. MCDP-II successfully detected several known fraudsters at time instants that coincided with major events on the Enron time-line. We observed that the majority of those fraudsters either changed behaviour in only one view, or exhibited different changes in different views. Nevertheless, MCDP-II was able to detect those subtle changes.

5.2 Future Work

Below we outline several possible future research directions that emerged from the work of this thesis.

Deciding the dimension of the embedding is one of the most critical steps of our method. An optimal low dimension, d, ensures that the embedded points amplify the connectivity structure of the graph and remove noise and redundant information. We adapted Achlioptas and McSherry (2007)'s low-rank matrix approximation method to determine d. From our discussions and experimental results, this method is a good alternative to the traditional scree-plot method in estimating the correct value for d especially in real-world graphs. However, when applied to large graphs, a drawback is the high computational cost associated with this method. It would be interesting to investigate other faster methods to estimate d in our algorithms. Jackson (1993) and Jolliffe (2002, Chapter 6) summarise several dimensionality selection methods that can be used for this purpose. However, as the truncation dimension, d, plays a major role in the accuracy of the change detection algorithm, careful investigation is required to understand the trade-off between accuracy and scalability in the selection of an alternative method.

In this thesis, we used generalized orthogonal Procrustes analysis techniques to calculate a profile embedding from the embeddings inside the window. If the window contains embedded points that are highly variable across time instants, the noise added by these points may prevent a change from being detected. To address this issue, it is possible to use a weighted Procrustes analysis procedure (Cootes et al., 1992). The weights can be selected to give higher importance to those points that are more stable and lesser importance to those points that have high variability inside the window. We believe that it would be worthwhile to investigate whether the use of weighted Procrustes analysis can improve change detection performance.

Our multi-view change detection methods, MCDP-I and MCDP-II, combine information from different slices of the tensor in order to obtain a single embedding of the multi-graph. During our simulation experiments, we investigated whether combining multiple slices is beneficial for change detection. We achieved this by comparing the performance of MCDP-I and MCDP-II with CDP applied on separate slices. When the change information is similar across slices, multi-view methods generally outperformed CDP. However, when there are discrepancies in the information contained in the slices, we observed several situations where multi-view methods showed worse performance than CDP on some slices. This shows that combining information from multiple views can be disadvantageous in the presence of conflicting information. Thus, we believe that it would be potentially beneficial to investigate the effect of identifying the relationship

between slices before combining them for change detection. For example, it is possible to find the correlation between slices and only combine those slices that are highly correlated. Also, some tensor decomposition methods (Liu et al., 2013) penalize noisy slices when obtaining factor matrices. It is possible to replace the HOSVD method in MCDP-I with one of these methods.

The scope of this thesis was confined to undirected graphs. However, many real-world graphs are directed and have asymmetric connections. Rohe et al. (2015) discuss a useful approach to obtain embeddings for vertices from a sparse and heterogeneous, directed graph represented by a matrix. For a directed multi-graph represented by a tensor, our HOSVD method can easily be extended to obtain embedded points for both senders and receivers. Since the embedded points are now of two different types, the extension of our GPA algorithm to obtain change scores is not straightforward. For example, one approach might be to introduce an additional weight parameter into the GPA objective function (Equation 3.10) to account for the two types of embedded points. Crosilla and Beinat (2002) and Gruen and Akca (2003) discuss some useful weighting strategies for GPA that can be employed for this purpose. Such an extension to directed graphs are also applicable to bipartite graphs where vertices are of two different types. Examples include graphs representing the relationships between documents and words, scientific papers and authors, users and products, and hosts and domains.

In this thesis, we mainly focused on developing a novel approach for change detection in dynamic networks. Additionally, it is possible to extend our developed method to perform anomaly detection. During our case studies involving the Enron email network, we discussed a simple thresholding method based on z-scores to find anomalies based on the distribution of the change scores from our algorithms. More sophisticated anomaly detection methods might be considered for use here instead of the z-scores. For example, recalling that we calculate the vertex change scores by normalizing the rows of the residual matrix resulting from the GPA, it is possible to perform diagnostic tests on this residual matrix to find the anomalies or anomalous regions in the graph. Dryden and Mardia (1998, Chapter 10) provide a collection of tests that can be considered for this purpose.

In summary, there is a high demand for efficient methods to detect changes in dynamic systems. By representing entities and their time-varying relationships as a time sequence of graphs, deviations in the evolving graph structures contain information about the underlying changes. The proposed approach in this thesis combines spectral embedding methods and Procrustes analysis techniques for vertex-based change detection. The

algorithms we have developed can be applied to a wide range of problems such as cyber-intrusion detection, fraud detection, healthcare monitoring and detection of natural disturbances in the eco-system.

Appendix A

Some Additional Proofs

In this Appendix we include some well known proofs of the properties of L_{rw} and L_{sym} stated in Sections 2.5.1.2 and 2.5.1.3, respectively.

1. λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$ if and only if λ and $\mathbf{x}$ solve the generalized eigenvalue problem, $L\mathbf{x} = \lambda D\mathbf{x}$.

   Proof. If λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$, we can write

   $$L_{rw}\mathbf{x} = \lambda\mathbf{x}.$$

   Substituting for L_{rw} from Equation 2.36,

   $$\Leftrightarrow D^{-1}L\mathbf{x} = \lambda\mathbf{x}.$$

   By multiplying both sides of the equation by D, we obtain

   $$\Leftrightarrow DD^{-1}L\mathbf{x} = D\lambda\mathbf{x}$$
   $$\Leftrightarrow L\mathbf{x} = \lambda D\mathbf{x}.$$

   □

2. λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$ if and only if $(1 - \lambda)$ is an eigenvalue of W_{rw} with eigenvector $\mathbf{x}$.

Proof. The relationship between L_{rw} and W_{rw} can be formulated as follows. From Equation 2.36,

$$\begin{aligned} L_{rw} &= D^{-1}L \\ &= D^{-1}(D - W) \quad \text{(substituting for } L \text{ from Equation 2.34)} \\ &= I - D^{-1}W \\ &= I - W_{rw}. \end{aligned} \tag{A.1}$$

Consider the eigen-equation of L_{rw},

$$L_{rw}\mathbf{x} = \lambda\mathbf{x}.$$

Substituting for L_{rw} from Equation A.1,

$$\Leftrightarrow (I - W_{rw})\mathbf{x} = \lambda\mathbf{x}.$$

Thus, we obtain,

$$\begin{aligned} \Leftrightarrow W_{rw}\mathbf{x} &= I\mathbf{x} - \lambda\mathbf{x} \\ &= (1 - \lambda)\mathbf{x}. \end{aligned} \tag{A.2}$$

□

3. λ is an eigenvalue of L_{rw} with eigenvector $\mathbf{x}$ if and only if λ is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbf{x}$.

Proof. Consider the eigen equation of L_{sym},

$$L_{sym}\mathbf{x} = \lambda\mathbf{x}.$$

By multiplying both sides of the equation by $D^{-1/2}$, we obtain

$$\Leftrightarrow D^{-1/2}L_{sym}\mathbf{x} = D^{-1/2}\lambda\mathbf{x}.$$

After substituting for L_{sym} from Equation 2.37, we have

$$\Leftrightarrow D^{-1}L(D^{-1/2}\mathbf{x}) = \lambda D^{-1/2}\mathbf{x}.$$

Finally, by substituting for $D^{-1}L$ from Equation 2.36, we have

$$\Leftrightarrow L_{rw}(D^{-1/2}\mathbf{x}) = \lambda D^{-1/2}\mathbf{x}.$$

□

Appendix B

Experiments Performed to Determine the Threshold of the Low-Rank Approximation Algorithm

In Section 3.2.3, we discussed how we use Algorithm 1 to determine the low-rank or truncation dimension d. There are two inputs to the algorithm, the input matrix and the threshold, ϵ, that determines the stopping criterion. In this appendix, we include the results of the experiments that were performed to determine ϵ, and also assess the performance of the algorithm. We refer to Algorithm 1 as DMR (dimension by matrix reconstruction), for brevity.

B.1 Simulation Experiments

We generate graphs using the same DCSBM model described in Section 3.3.2. Keeping other parameters constant, we vary the number of blocks, in order to give rise to different truncation dimensions. The other fixed parameters are defined as for M1 in Table 3.1.

For each generated graph, we construct M and use as input to DMR. We do not define a threshold, and let the algorithm run for 100 iterations. Recall our measure obtained at each iteration k from Algorithm 1 which is,

$$\rho_k = \frac{|\parallel R_k \parallel_2 - \parallel \tilde{R}_k \parallel_2 |}{\parallel R_k \parallel_F}, \tag{B.1}$$

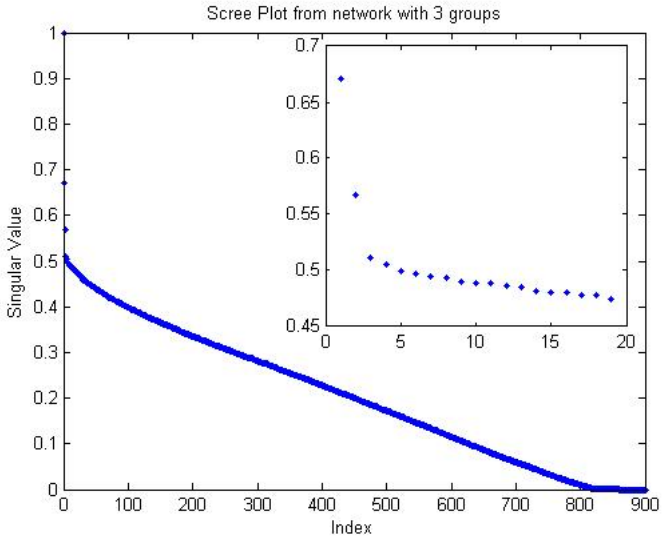


FIGURE B.1: The scree plot of the singular values from the pre processed representation matrix from a graph with 3 blocks. The singular values from 2nd onwards is magnified at the top right corner.

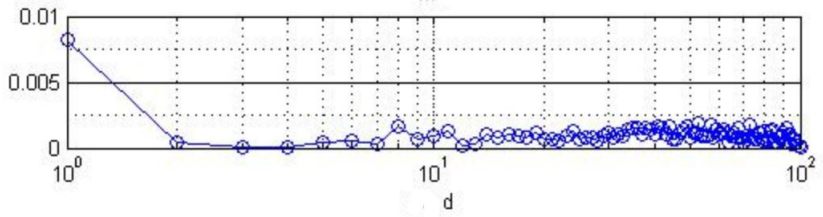


FIGURE B.2: The semi-log plot showing the pattern followed by ρ for the same graph over 100 iterations of the DMR algorithm.

where $\| \cdot \|_2$ denotes the L_2 norm, $|.|$ denotes absolute value. We observe the pattern followed by ρ_k for $k = 1, 2, \ldots, 100$, and compare it with the scree plot of M.

In Figure B.1, we observe the decreasing trend of the scree plot for a graph with 3 blocks. The magnified plot at the top right corner depicts the scree from second index onwards. The scree plot indicates a clear cut off at 2 singular values starting from the second singular value. In figure B.2, we observe the pattern followed by ρ_k for $k = 1, 2, \ldots, 100$ for the same graph. We see a sharp decrease of ρ_k for the transition from $k = 1$ to $k = 2$, i.e., from the rank-2 reconstruction to the rank-3 reconstruction of the matrix. From $k = 2$ onwards we do not observe a major decrease in ρ_k. For this 3-block graph, a threshold of approximately 0.005 returns the correct value for d in DMR.

Similarly, Figure B.3 shows the scree plot for a graph with 4 blocks. The scree plot indicates a cut off at 2 or 3 singular values starting from the second singular value. In figure B.4, we observe the pattern followed by ρ_k for $k = 1, 2, \ldots, 100$ for the same graph. We see a sharp decrease of ρ_k from the rank-2 reconstruction to the rank-3 reconstruction

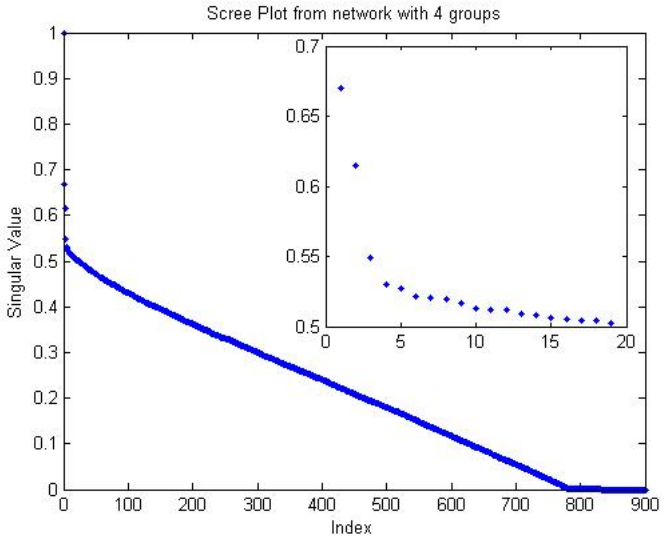


FIGURE B.3: The scree plot of the singular values from the pre processed representation matrix from a graph with 4 blocks. The singular values from 2nd onwards is magnified at the top right corner.

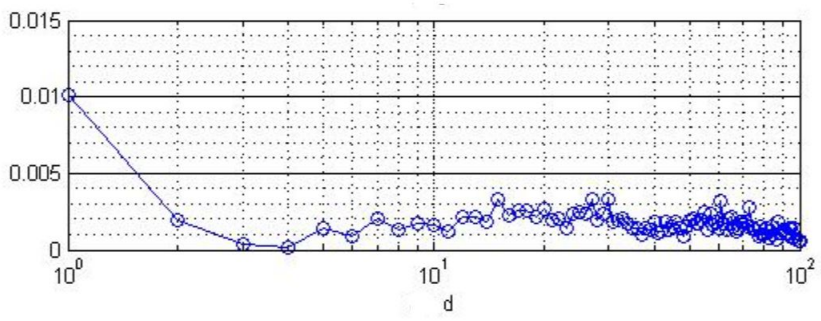


FIGURE B.4: The semi-log plot showing the pattern followed by ρ for the same graph over 100 iterations of the DMR algorithm

of the matrix, and a bit lower decrease in ρ_k from the rank-3 reconstruction to the rank-4 reconstruction. From rank-4 reconstruction onwards, we do not observe a major decrease in ρ_k. For this 4-block graph, a threshold of approximately $0.005 - 0.003$ returns $d = 2$.

In a similar manner Figures B.5 and B.7 show the scree plots for graphs generated with 6 and 9 blocks respectively. The corresponding plots of ρ_k for these graphs are given in Figures B.6 and B.8 respectively. We observe how both results indicate a threshold of 0.005 to be satisfactory to obtain a value for d, that agrees with the value returned from the scree plot.

From the simple simulation experiments performed, we observe that DMR with threshold at $\epsilon = 0.005$, can be used to obtain a good estimate of the suitable truncation dimension d.

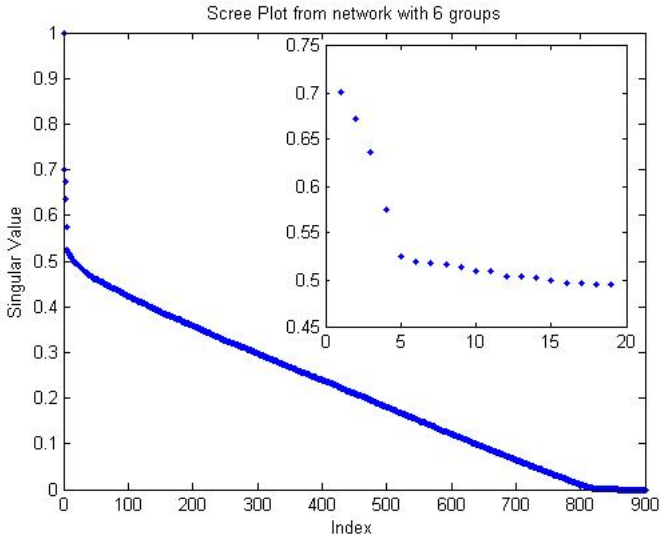


FIGURE B.5: The scree plot of the singular values from the pre processed representation matrix from a graph with 6 blocks. The singular values from 2nd onwards is magnified at the top right corner.

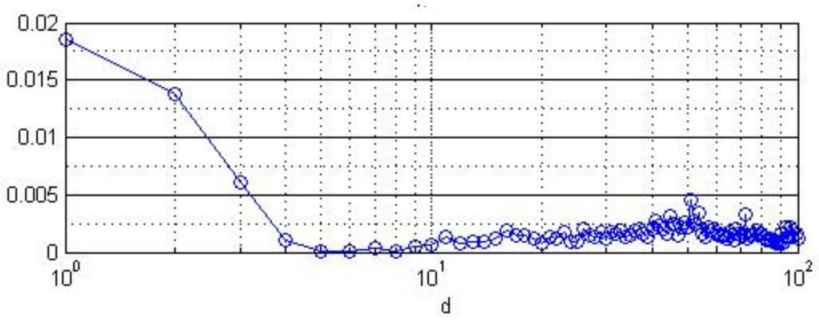


FIGURE B.6: The semi-log plot showing the pattern followed by ρ for the same graph over 100 iterations of the DMR algorithm

B.2 Experiment on Real Data

The simulation experiments in Section B.1 show the suitability of $\epsilon = 0.005$ as the stopping criteria of DMR for correct dimension selection. We further apply DMR on a real-world dataset, where the graph structure is more random and noisier than simulated graphs.

The network we study is obtained from the terrorist attack dataset from the Collective-Classification-Database. The network consists of $1,293$ different terrorist attacks that are labelled under six categories as *arson, bombing, kidnapping, NBCR attack, weapon attack* and *other attack*. The attacks are connected to each other if they were planned

by the same organization. We obtain an undirected, unweighted graph from the dataset. In figure B.9 we observe the scree plot of M obtained from the graph. Figure B.10 shows the trend of ρ_k by allowing the DMR algorithm to iterate for 100 runs. We observe that although it is difficult to make a clear decision using the scree plot, DMR algorithm with $\epsilon = 0.005$, would give an estimate as $d = 8$ for this situation.

In summary, our experiments indicate a threshold of 0.005 to be suitable as the input ϵ for the DMR Algorithm. Both simulation and real-world experiments depict DMR to be as reliable as the scree plot and sometimes even better to estimate the low-rank or truncation dimension, d, of a matrix.

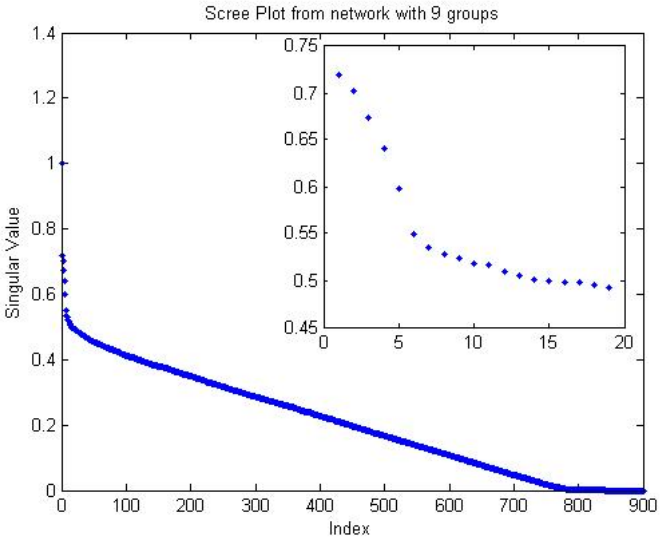


FIGURE B.7: The scree plot of the singular values from the pre processed representation matrix from a graph with 9 blocks. The singular values from 2nd onwards is magnified at the top right corner.

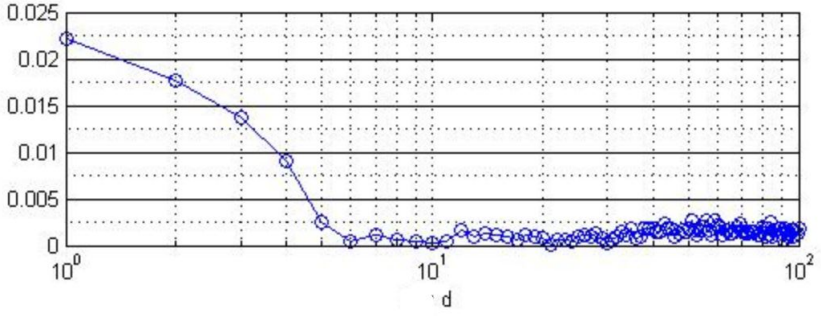


FIGURE B.8: The semi-log plot showing the pattern followed by ρ for the same graph over 100 iterations of the DMR algorithm

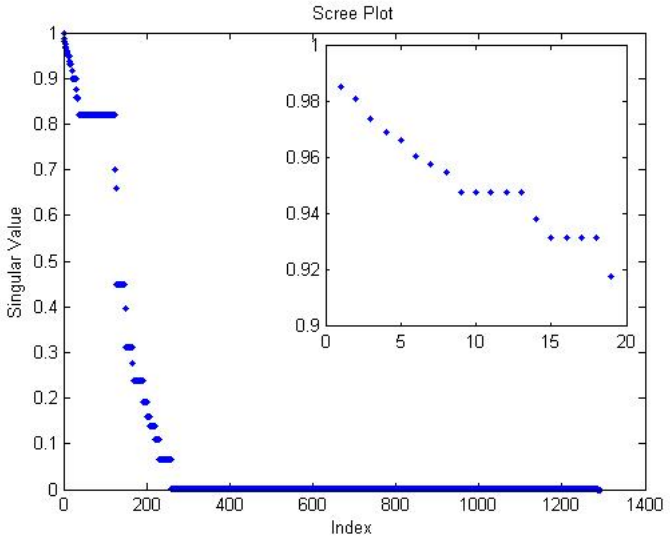


FIGURE B.9: The scree plot of the singular values from the pre processed representation matrix from the terrorist attack graph. The singular values from 2nd onwards is magnified at the top right corner.

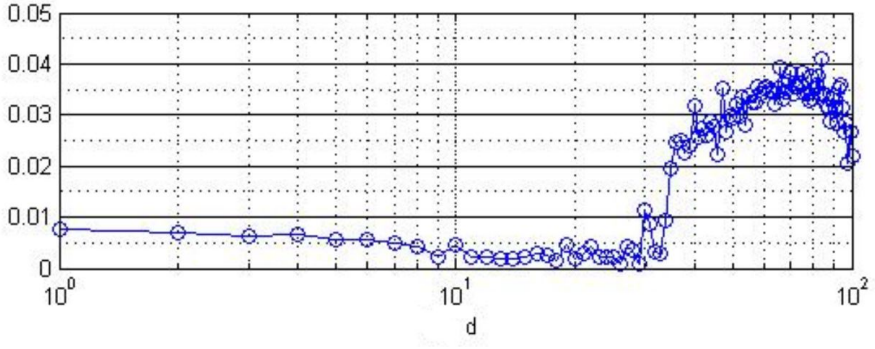


FIGURE B.10: The semi-log plot showing the pattern followed by ρ for the same graph over 100 iterations of the DMR algorithm

Appendix C

Change Detection in Dynamic Networks - Additional Results

This appendix contains some additional results from Section 3.3.6 of Chapter 3. We provide our results on the variation of $\boldsymbol{\eta}^t$ for change scenarios, split (Figure C.1), merge (Figure C.2), form (Figure C.3), fragment (Figure C.4), hetero-to-homo (Figure C.5), homo-to-hetero (Figure C.6), simple-to-complex (Figure C.7), and complex-to-simple (Figure C.8).

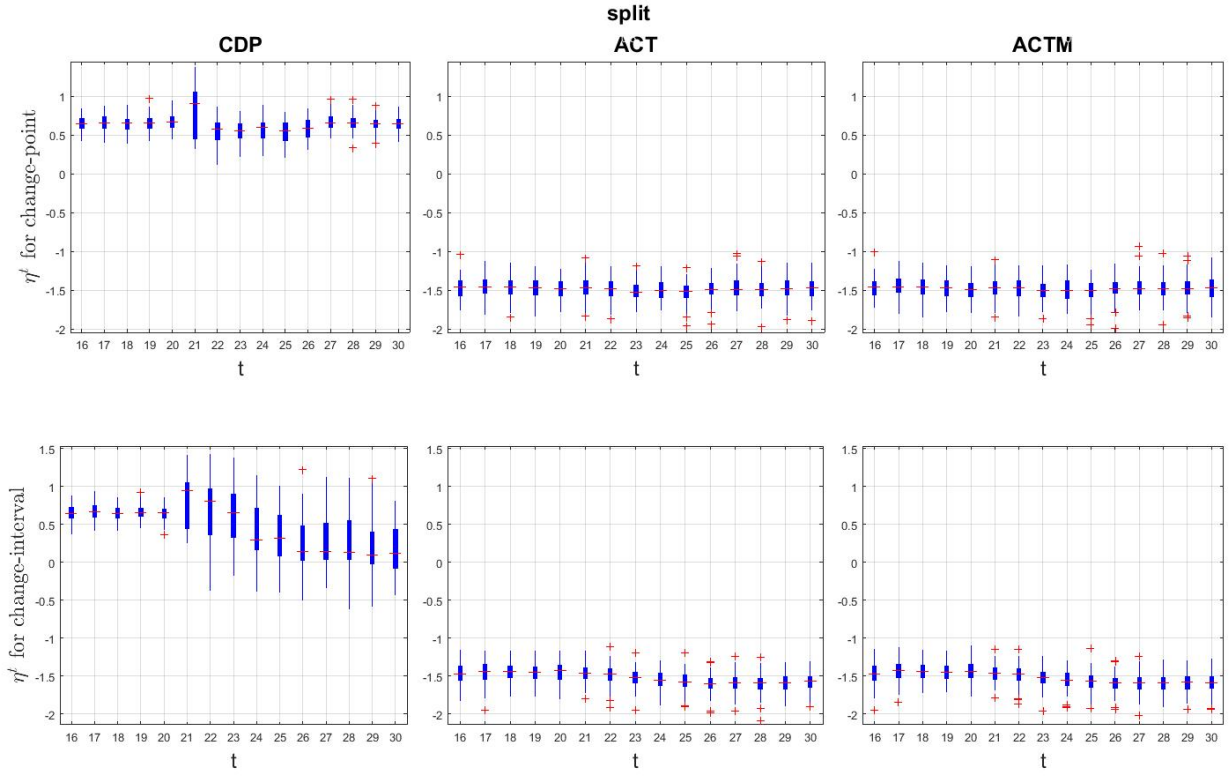


FIGURE C.1: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on split for $w = 5$ change point(top) and change-interval (bottom).

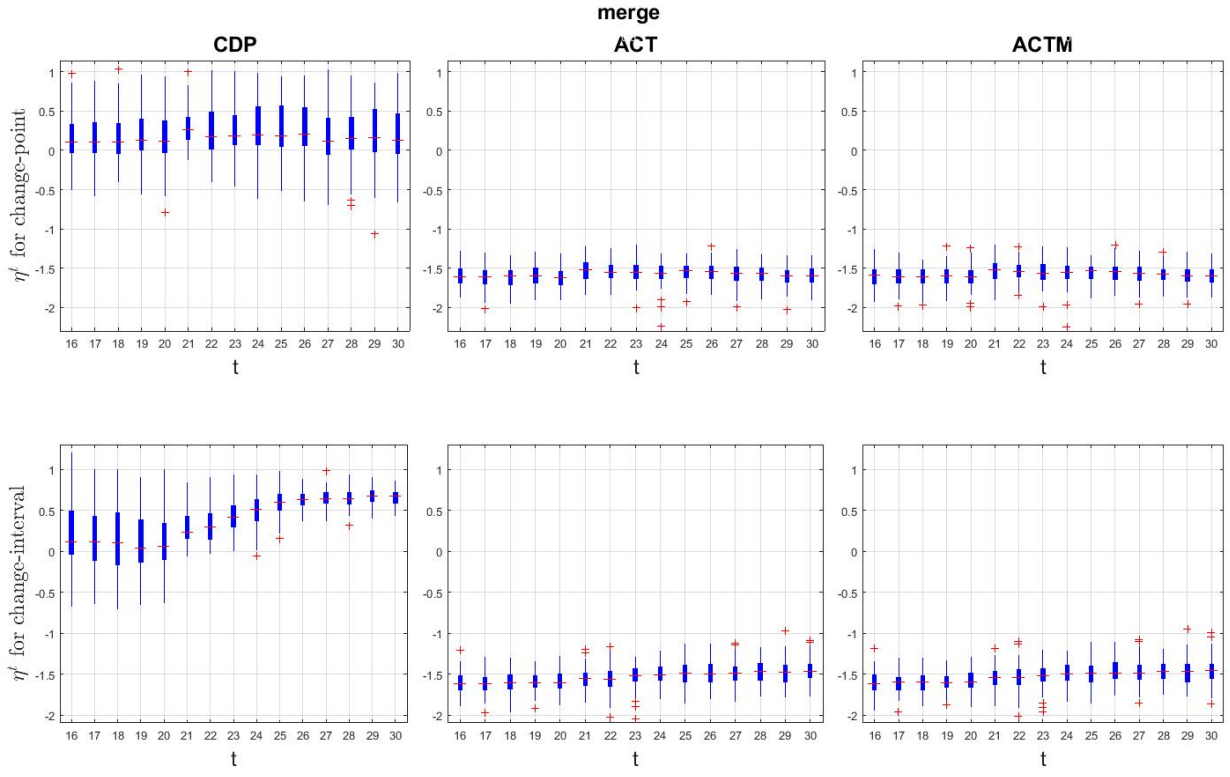


FIGURE C.2: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on merge for $w = 5$ change point(top) and change-interval (bottom).

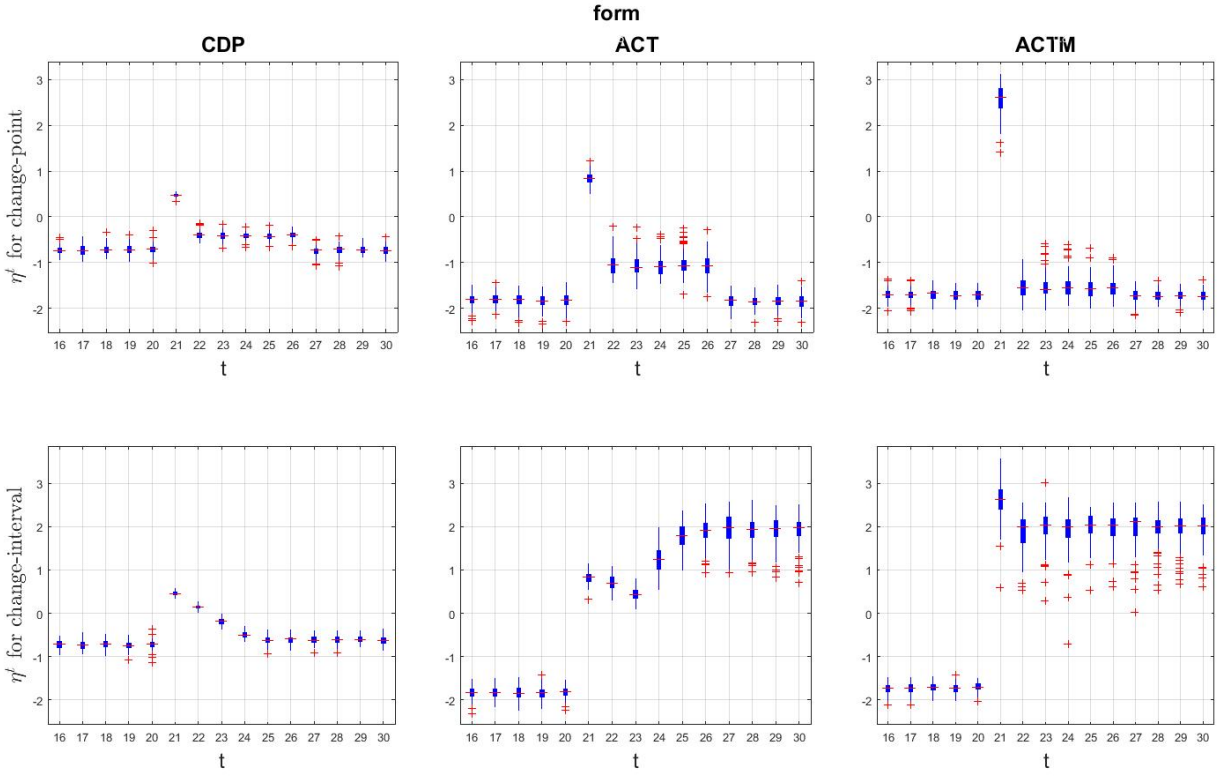


FIGURE C.3: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on form for $w = 5$ change point(top) and change-interval (bottom).

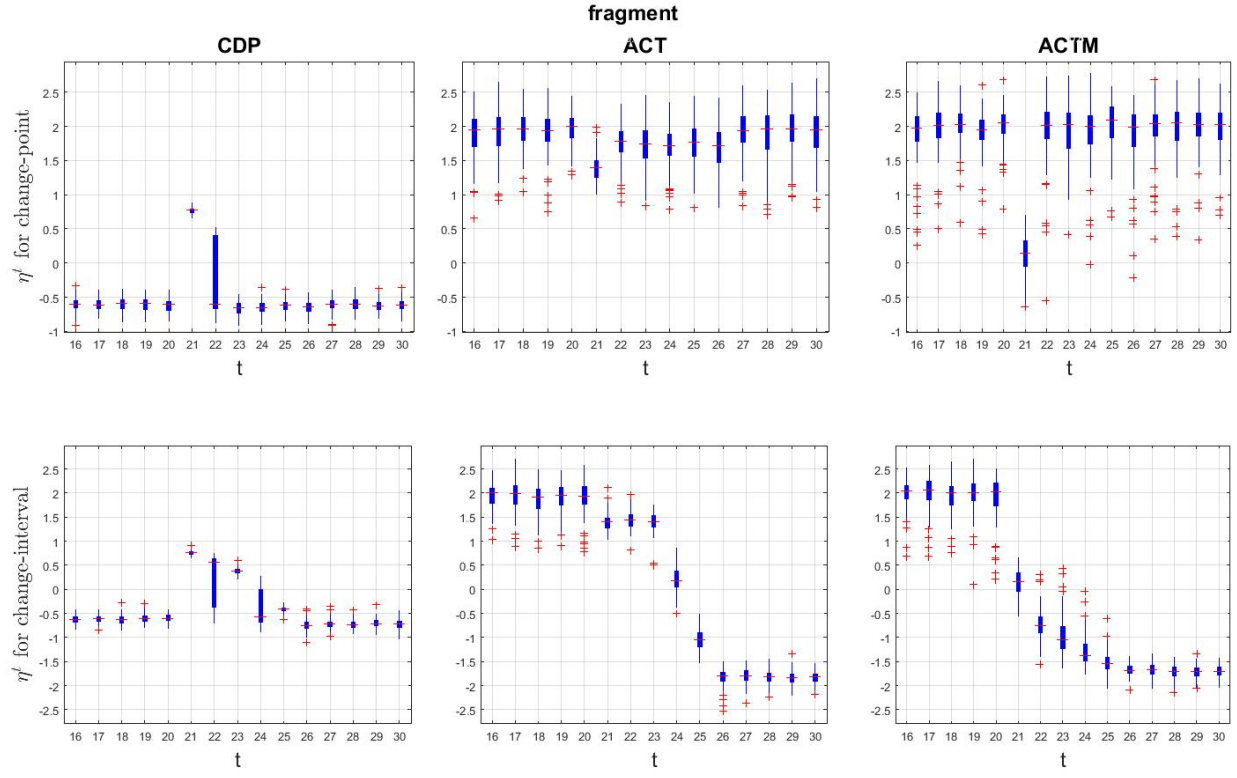


FIGURE C.4: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on fragment for $w = 5$ change point(top) and change-interval (bottom).

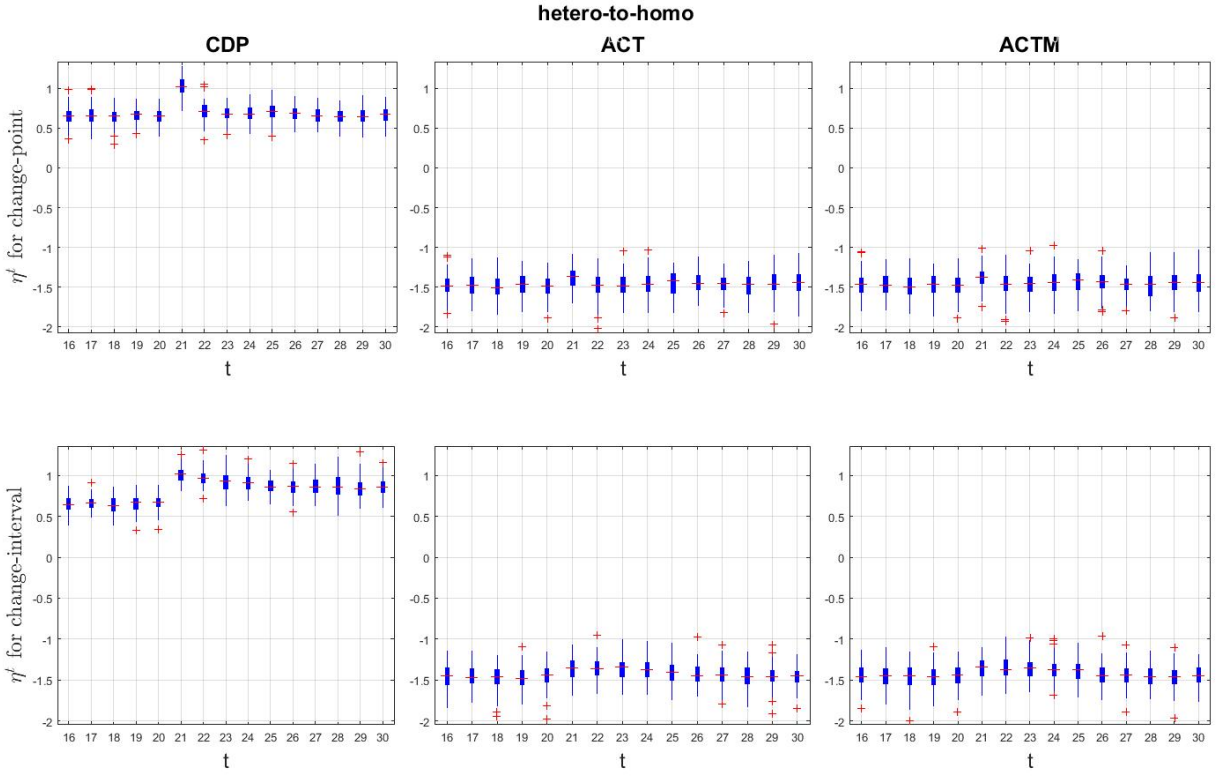


FIGURE C.5: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on hetero-to-homo for $w = 5$ change point(top) and change-interval (bottom).

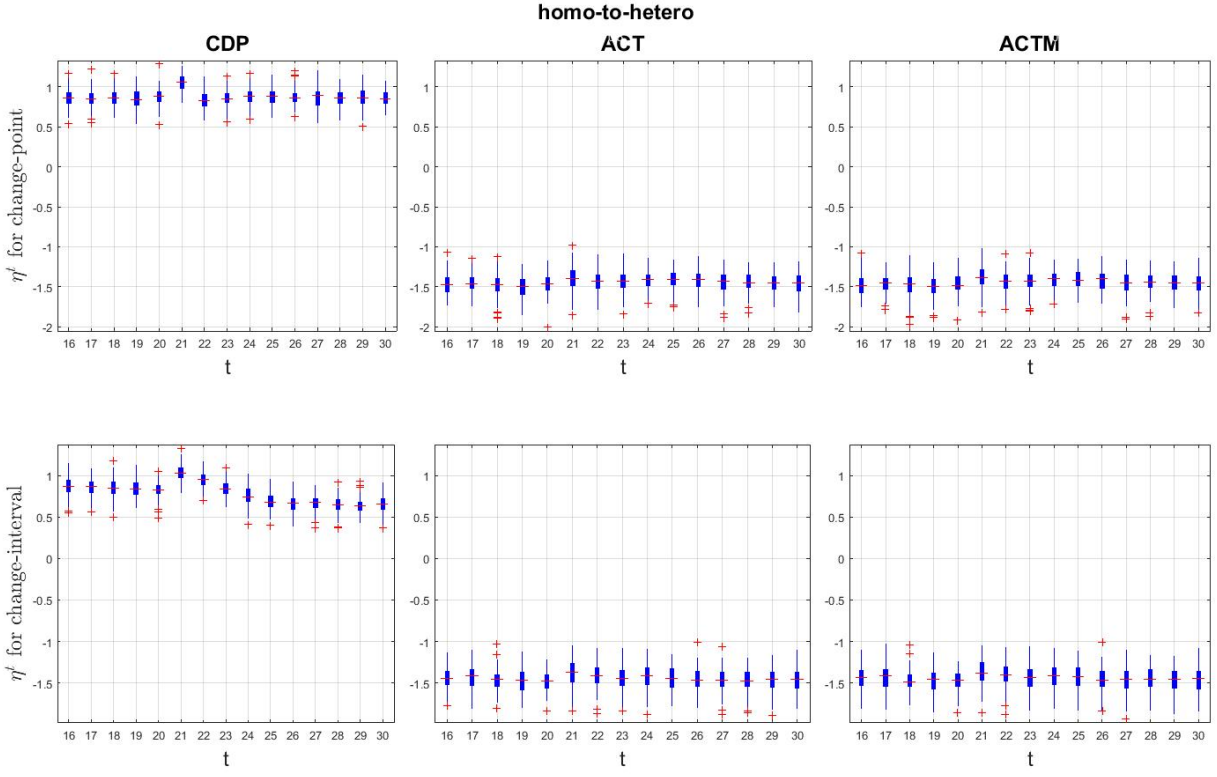


FIGURE C.6: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on homo-to-hetero for $w = 5$ change point(top) and change-interval (bottom).

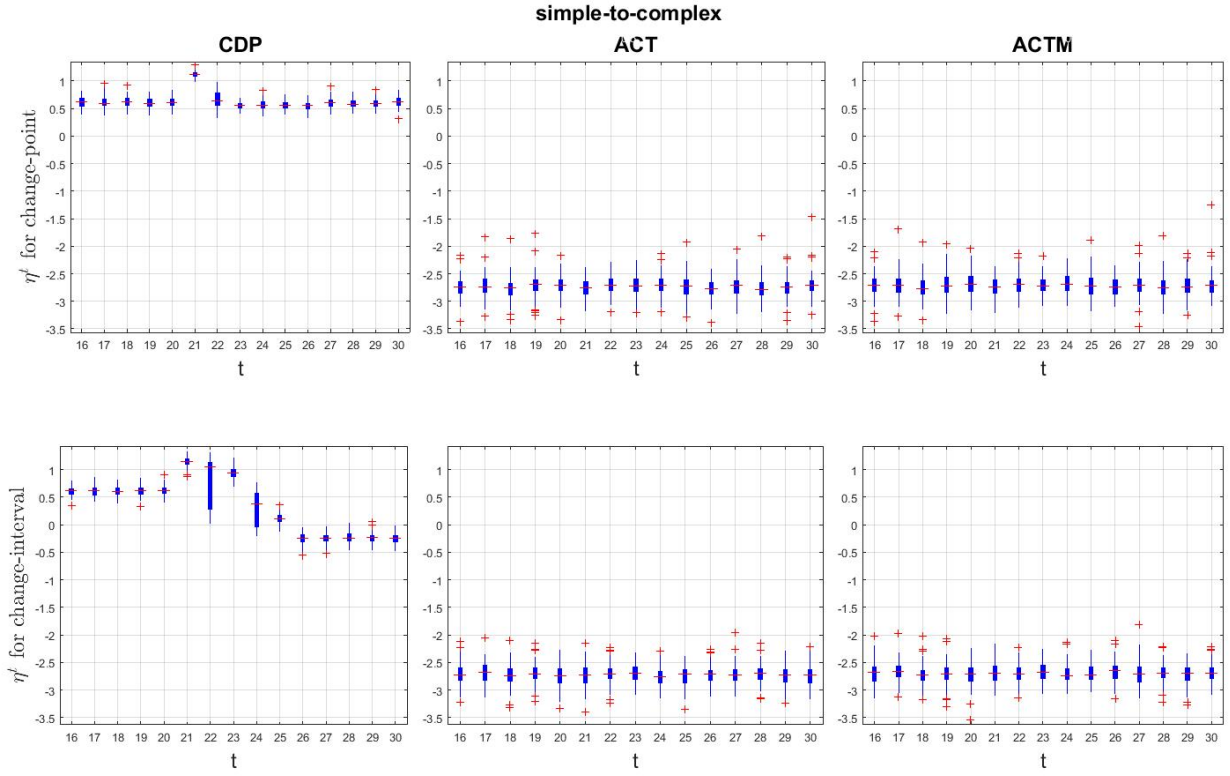


FIGURE C.7: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on simple-to-complex for $w = 5$ change point(top) and change-interval (bottom).

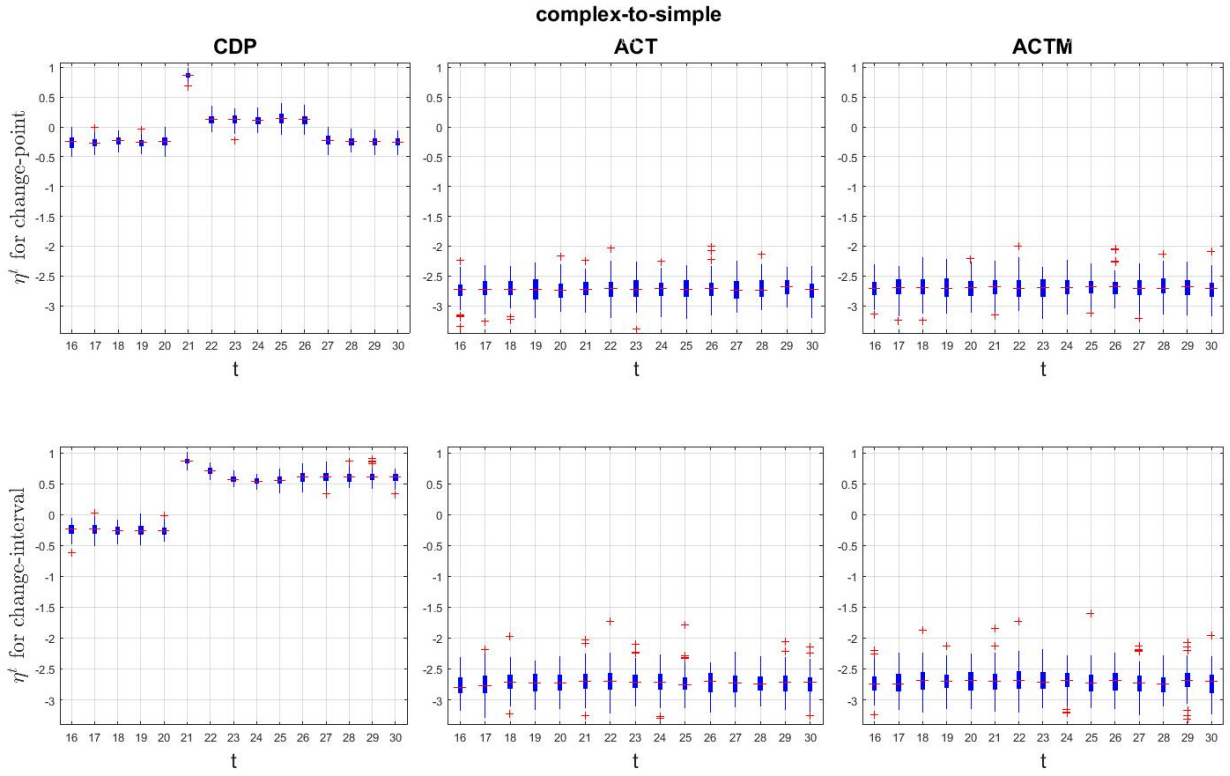


FIGURE C.8: Observing $\boldsymbol{\eta}^t$ on CDP (left), ACT (middle), and ACTM (right) along time on complex-to-simple for $w = 5$ change point(top) and change-interval (bottom).

Appendix D

Change Detection in Multi-view Networks-Additional Results

D.1 Illustration of Change Scenarios

In this section, we provide an illustration of the change scenarios described in Section 4.2. The change each slice undergoes in each scenario can be clearly seen in the given figures.

1. *group-change*: Vertices 150:300 change membership from block one to block two. Due to this reason, block one shrinks and block two expands, leading to a change in the behaviour of vertices 1:600.

2. *hetero-to-homo*: Recall from Section 3.3.2, that each slice is generated from a degree corrected stochastic block model. Thus, the degree distribution of vertices is heterogeneous. In this change scenario, vertices 1:300 (consisting of block one) change their degree distribution to a homogeneous degree distribution, that is, there is a change in the degree of the latter set of vertices.

3. *fragment*: This change scenario corresponds to vertices 601:900 loosing the edges between them, that is, block three is very sparsely connected after the change.

4. *mixed*: In this scenario, we generate a multi-graph network only in the complex situation and let vertices in different slices under-go different change scenario:

    - In slice–1 and slice–2, vertices 1:300 (block one) change their degree distribution from heterogeneous to homogeneous (hetero-to-homo scenario).
    - In blocks one and two of slice–3, vertices 1:600 lose all their inter-block edges (complex-to-simple scenario of Section 3.3.3).

- In slice–4 vertices 1:300 change their degrees from heterogeneous to homogeneous (hetero-to-homo scenario), while vertices 301:600 form edges between them to become a tightly connected community (form scenario).

Considering the changes occurring in all four slices,

- vertices 1:300 change behaviour in all slices (*All*) and
- vertices 301:600 change behaviour in only two slices (*Two*).

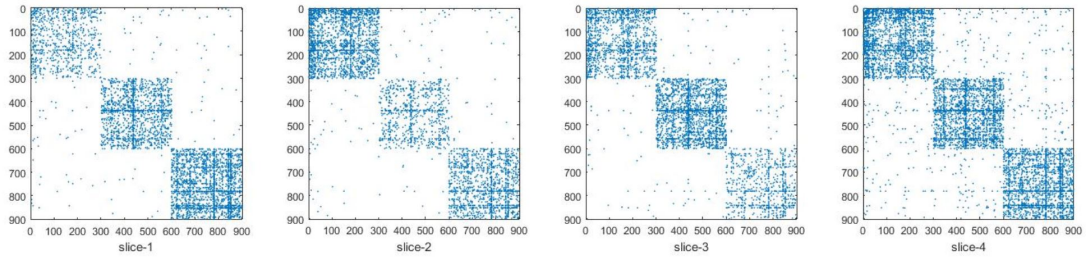


FIGURE D.1: F_0 in simple

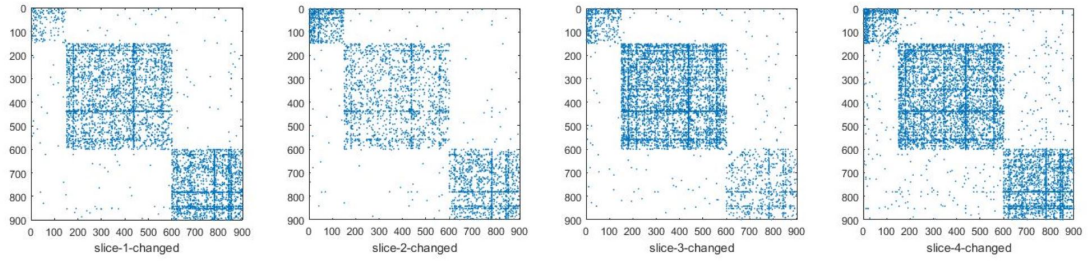


FIGURE D.2: F_1 for group-change in simple

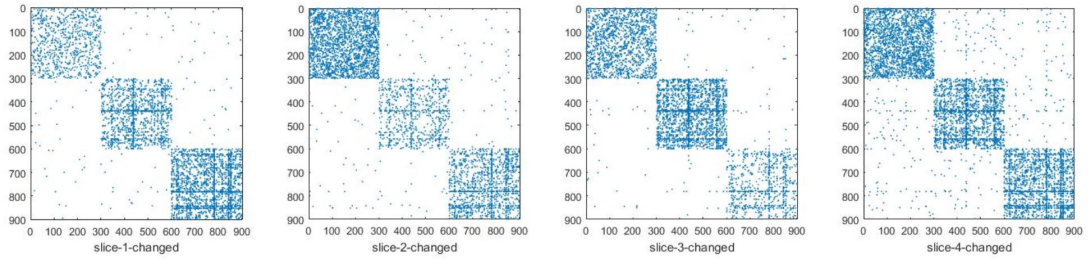


FIGURE D.3: F_1 for hetero-to-homo in simple

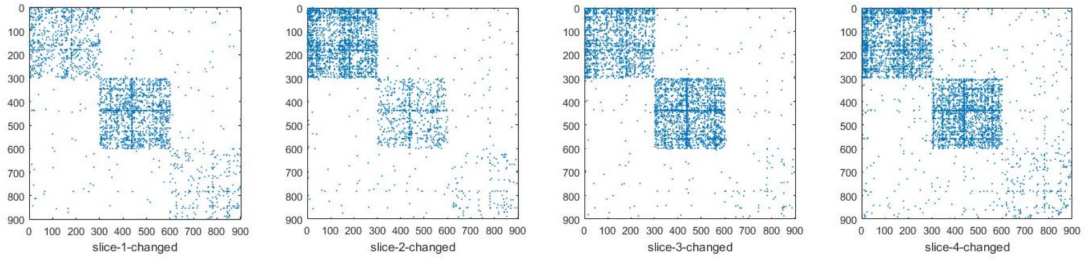


FIGURE D.4: F_1 for fragment in simple

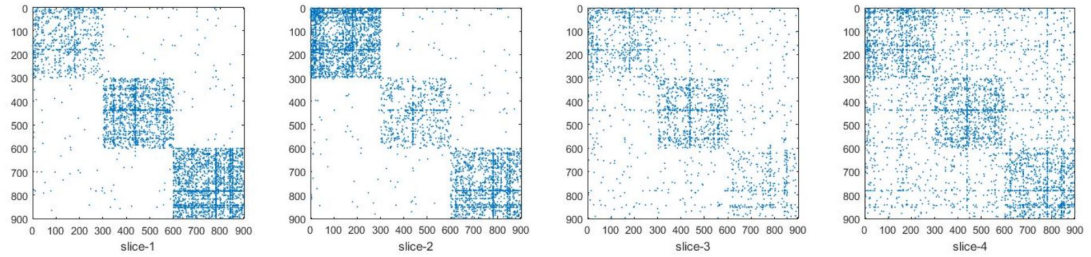


FIGURE D.5: F_0 in noisy

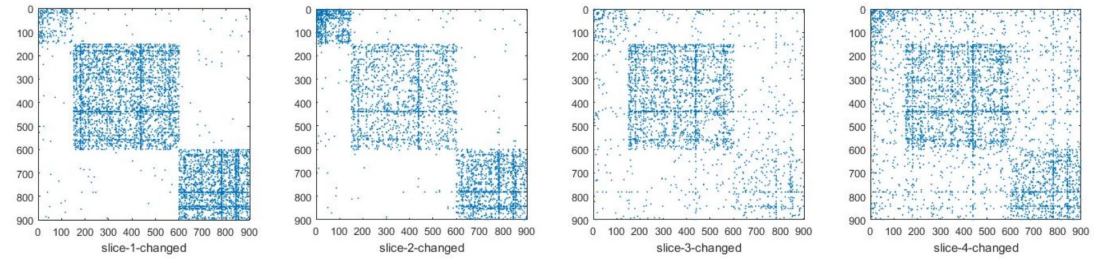


FIGURE D.6: F_1 for group-change in noisy

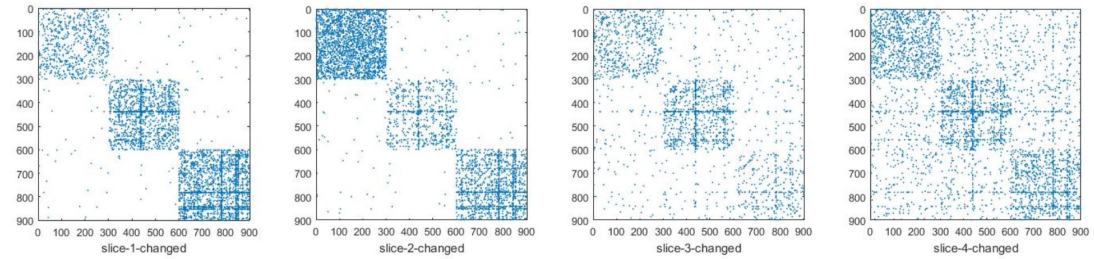


FIGURE D.7: F_1 for hetero-to-homo in noisy

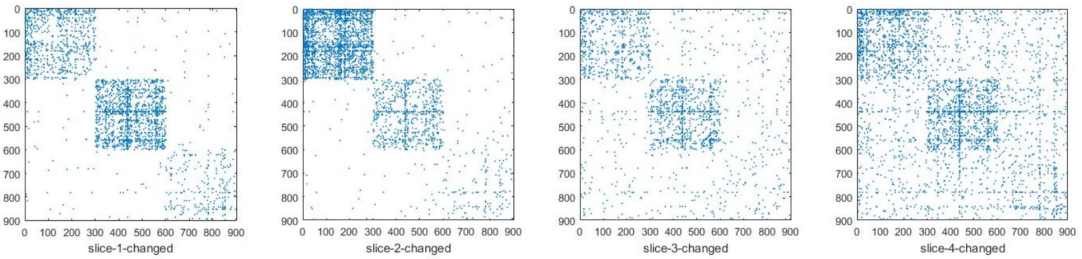


FIGURE D.8: F_1 for fragment in noisy

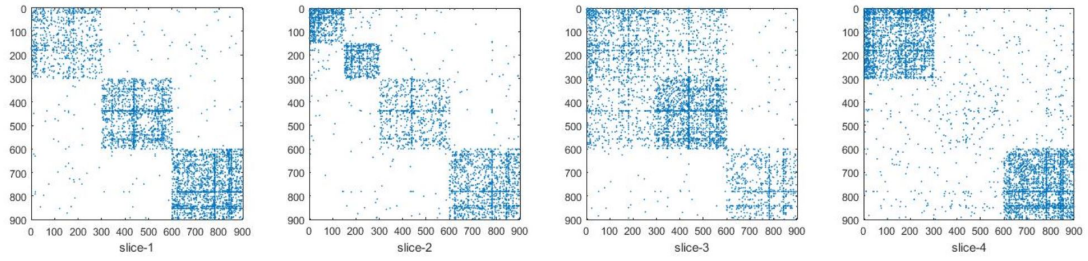


FIGURE D.9: F_0 in complex

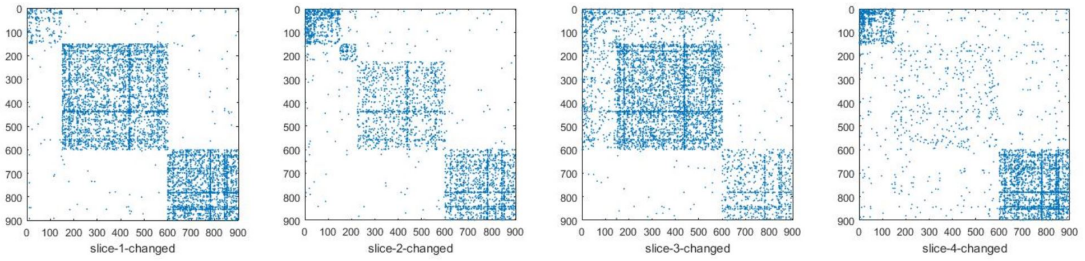


FIGURE D.10: F_1 for group-change in complex

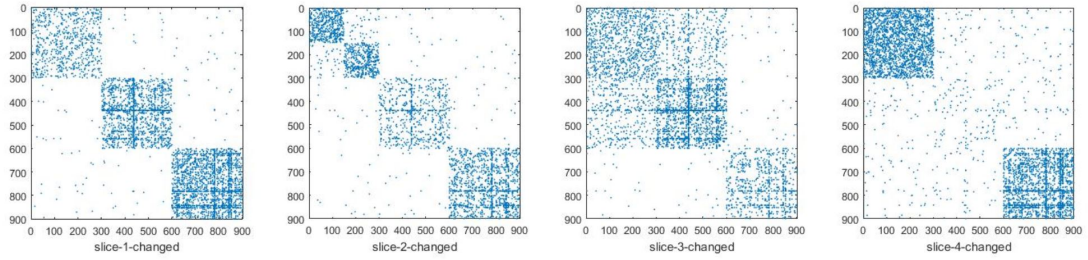


FIGURE D.11: F_1 for hetero-to-homo in complex

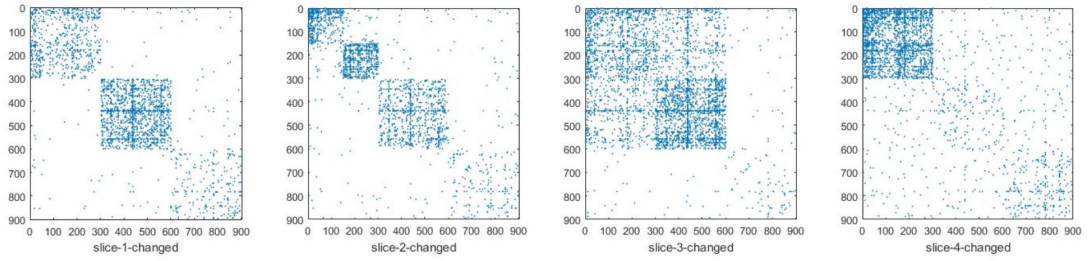


FIGURE D.12: F_1 for fragment in complex

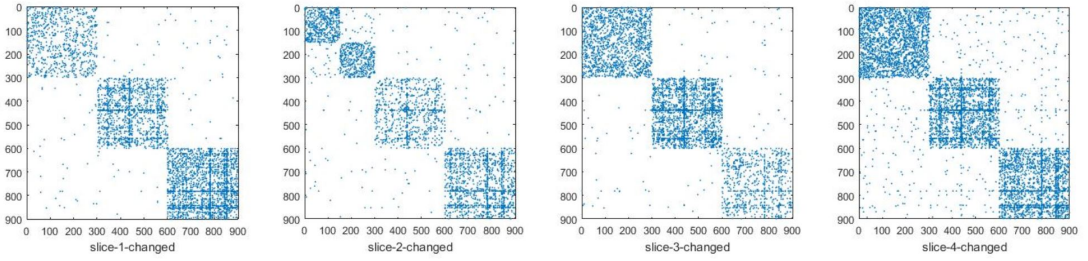


FIGURE D.13: F_1 for mixed in complex

D.2 Statistical Test Results

D.2.1 Simple Model

The sign test p-values given in Table D.1, and corresponding proportions in Table D.2 show the statistical significance of the results in the simple situation for group-change (illustrated in Figure 4.6), fragment (illustrated in Figure 4.7) and hetero-to-homo (illustrated in Figure 4.8) of Section 4.2.2.

D.2.2 Noisy Model

The sign test p-values given in Table D.3, and corresponding proportions in Table D.4 show the statistical significance of the results in the noisy situation for group-change (illustrated in Figure 4.6), fragment (illustrated in Figure 4.7) and hetero-to-homo (illustrated in Figure 4.8) of Section 4.2.2.

D.2.3 Complex Model

The sign test p-values given in Table D.5, and corresponding proportions in Table D.6 show the statistical significance of the results in the complex situation for group-change (illustrated in Figure 4.6), fragment (illustrated in Figure 4.7) and hetero-to-homo (illustrated in Figure 4.8) of Section 4.2.2.

D.2.4 Mixed-Change

The sign test p-values given in Table D.7, and corresponding proportions in Table D.8 show the statistical significance of the results illustrated in Figure 4.9 for mixed-change scenario of Section 4.2.2.

D.3 Performance Comparison for Several Time Instants Before and After Change

In this section, we show the results of performance measure $\bar{\boldsymbol{\eta}}^t$ for several time instants before and after the change point, for change scenarios group-change (illustrated in Figure 4.6), fragment (illustrated in Figure 4.7), hetero-to-homo (illustrated in Figure 4.8) and mixed-change (illustrated in Figure 4.9) discussed in Section 4.2.2.

TABLE D.1: Sign test p-values in simple situation

	group-change	fragment	hetero-to-homo
MCDP-I > MCDP-II	0.00000	0.00000	0.00000
MCDP-I < MCDP-II	1.00000	1.00000	1.00000
MCDP-I ≠ MCDP-II	0.00000	0.00000	0.00000
MCDP-I > AVG	1.00000	1.00000	1.00000
MCDP-I < AVG	0.00000	0.00000	0.00000
MCDP-I ≠ AVG	0.00000	0.00000	0.00000
MCDP-I > MAX	1.00000	1.00000	0.00347
MCDP-I < MAX	0.00000	0.00000	0.99813
MCDP-I ≠ MAX	0.00000	0.00000	0.00693
MCDP-I > AVGU	1.00000	0.00000	0.00000
MCDP-I < AVGU	0.00000	1.00000	1.00000
MCDP-I ≠ AVGU	0.00000	0.00000	0.00000
MCDP-I > WSUMU	1.00000	0.00000	0.00000
MCDP-I < WSUMU	0.00000	1.00000	1.00000
MCDP-I ≠ WSUMU	0.00000	0.00000	0.00000
MCDP-II > AVG	1.00000	1.00000	1.00000
MCDP-II < AVG	0.00000	0.00000	0.00000
MCDP-II ≠ AVG	0.00000	0.00000	0.00000
MCDP-II > MAX	1.00000	1.00000	1.00000
MCDP-II < MAX	0.00000	0.00000	0.00000
MCDP-II ≠ MAX	0.00000	0.00000	0.00000
MCDP-II > AVGU	1.00000	0.00000	1.00000
MCDP-II < AVGU	0.00000	1.00000	0.00000
MCDP-II ≠ AVGU	0.00000	0.00000	0.00000
MCDP-II > WSUMU	1.00000	0.00000	1.00000
MCDP-II < WSUMU	0.00000	1.00000	0.00000
MCDP-II ≠ WSUMU	0.00000	0.00000	0.00000
AVG > MAX	1.00000	1.00000	0.00000
AVG < MAX	0.00000	0.00000	1.00000
AVG ≠ MAX	0.00000	0.00000	0.00000
AVG > AVGU	0.00000	0.00000	0.00000
AVG < AVGU	1.00000	1.00000	1.00000
AVG ≠ AVGU	0.00000	0.00000	0.00000
AVG > WSUMU	0.00000	0.00000	0.00000
AVG < WSUMU	1.00000	1.00000	1.00000
AVG ≠ WSUMU	0.00000	0.00000	0.00000
MAX > AVGU	0.00000	0.00000	0.00621
MAX < AVGU	1.00000	1.00000	0.99653
MAX ≠ AVGU	0.00000	0.00000	0.01242
MAX > WSUMU	0.00000	0.00000	0.00097
MAX < WSUMU	1.00000	1.00000	0.99952
MAX ≠ WSUMU	0.00000	0.00000	0.00194
AVGU > WSUMU	0.93319	0.99977	0.86433
AVGU < WSUMU	0.09680	0.00048	0.18406
AVGU ≠ WSUMU	0.19360	0.00097	0.36812

Table D.2: Sign test proportions in simple situation

	group-change	fragment	hetero-to-homo
MCDP-I > MCDP-II	1.0000	0.7500	1.0000
MCDP-I < MCDP-II	0.0000	0.2500	0.0000
MCDP-I ≠ MCDP-II	1.0000	1.0000	1.0000
MCDP-I > AVG	0.0000	0.1000	0.0800
MCDP-I < AVG	1.0000	0.9000	0.9200
MCDP-I ≠ AVG	1.0000	1.0000	1.0000
MCDP-I > MAX	0.0000	0.0400	0.6400
MCDP-I < MAX	1.0000	0.9600	0.3600
MCDP-I ≠ MAX	1.0000	1.0000	1.0000
MCDP-I > AVGU	0.1100	0.9500	0.8800
MCDP-I < AVGU	0.8900	0.0500	0.1200
MCDP-I ≠ AVGU	1.0000	1.0000	1.0000
MCDP-I > WSUMU	0.1000	0.9600	0.8600
MCDP-I < WSUMU	0.9000	0.0400	0.1400
MCDP-I ≠ WSUMU	1.0000	1.0000	1.0000
MCDP-II > AVG	0.0000	0.0700	0.0100
MCDP-II < AVG	1.0000	0.9300	0.9900
MCDP-II ≠ AVG	1.0000	1.0000	1.0000
MCDP-II > MAX	0.0000	0.0200	0.1300
MCDP-II < MAX	1.0000	0.9800	0.8700
MCDP-II ≠ MAX	1.0000	1.0000	1.0000
MCDP-II > AVGU	0.0200	0.7600	0.1500
MCDP-II < AVGU	0.9800	0.2400	0.8500
MCDP-II ≠ AVGU	1.0000	1.0000	1.0000
MCDP-II > WSUMU	0.0100	0.7600	0.1400
MCDP-II < WSUMU	0.9900	0.2400	0.8600
MCDP-II ≠ WSUMU	1.0000	1.0000	1.0000
AVG > MAX	0.0100	0.1200	0.9900
AVG < MAX	0.9900	0.8800	0.0100
AVG ≠ MAX	1.0000	1.0000	1.0000
AVG > AVGU	0.9800	0.9800	1.0000
AVG < AVGU	0.0200	0.0200	0.0000
AVG ≠ AVGU	1.0000	1.0000	1.0000
AVG > WSUMU	0.9800	1.0000	1.0000
AVG < WSUMU	0.0200	0.0000	0.0000
AVG ≠ WSUMU	1.0000	1.0000	1.0000
MAX > AVGU	1.0000	0.9800	0.6300
MAX < AVGU	0.0000	0.0200	0.3700
MAX ≠ AVGU	1.0000	1.0000	1.0000
MAX > WSUMU	1.0000	0.9900	0.6600
MAX < WSUMU	0.0000	0.0100	0.3400
MAX ≠ WSUMU	1.0000	1.0000	1.0000
AVGU > WSUMU	0.4300	0.3300	0.4500
AVGU < WSUMU	0.5700	0.6700	0.5500
AVGU ≠ WSUMU	1.0000	1.0000	1.0000

TABLE D.3: Sign test p-values in noisy situation

	group-change	fragment	hetero-to-homo
MCDP-I > MCDP-II	0.00000	0.00347	0.00000
MCDP-I < MCDP-II	1.00000	0.99813	1.00000
MCDP-I ≠ MCDP-II	0.00000	0.00693	0.00000
MCDP-I > AVG	1.00000	0.00000	0.00000
MCDP-I < AVG	0.00000	1.00000	1.00000
MCDP-I ≠ AVG	0.00000	0.00000	0.00000
MCDP-I > MAX	1.00000	0.00347	0.00000
MCDP-I < MAX	0.00000	0.99813	1.00000
MCDP-I ≠ MAX	0.00000	0.00693	0.00000
MCDP-I > AVGU	0.06681	0.00000	0.00000
MCDP-I < AVGU	0.95543	1.00000	1.00000
MCDP-I ≠ AVGU	0.13361	0.00000	0.00000
MCDP-I > WSUMU	1.00000	0.00000	0.00000
MCDP-I < WSUMU	0.00000	1.00000	1.00000
MCDP-I ≠ WSUMU	0.00000	0.00000	0.00000
MCDP-II > AVG	1.00000	0.00000	1.00000
MCDP-II < AVG	0.00000	1.00000	0.00000
MCDP-II ≠ AVG	0.00000	0.00000	0.00000
MCDP-II > MAX	1.00000	0.13567	0.99999
MCDP-II < MAX	0.00000	0.90320	0.00002
MCDP-II ≠ MAX	0.00000	0.27133	0.00004
MCDP-II > AVGU	1.00000	0.00000	1.00000
MCDP-II < AVGU	0.00000	1.00000	0.00000
MCDP-II ≠ AVGU	0.00000	0.00000	0.00000
MCDP-II > WSUMU	1.00000	0.00000	1.00000
MCDP-II < WSUMU	0.00000	1.00000	0.00000
MCDP-II ≠ WSUMU	0.00000	0.00000	0.00000
AVG > MAX	0.95543	1.00000	0.00000
AVG < MAX	0.06681	0.00000	1.00000
AVG ≠ MAX	0.13361	0.00000	0.00000
AVG > AVGU	0.00000	0.00000	0.00000
AVG < AVGU	1.00000	1.00000	1.00000
AVG ≠ AVGU	0.00000	0.00000	0.00000
AVG > WSUMU	0.00000	0.00000	0.00000
AVG < WSUMU	1.00000	1.00000	1.00000
AVG ≠ WSUMU	0.00000	0.00000	0.00000
MAX > AVGU	0.00000	0.00000	0.99379
MAX < AVGU	1.00000	1.00000	0.01072
MAX ≠ AVGU	0.00000	0.00000	0.02145
MAX > WSUMU	0.00000	0.00000	1.00000
MAX < WSUMU	1.00000	1.00000	0.00000
MAX ≠ WSUMU	0.00000	0.00000	0.00000
AVGU > WSUMU	1.00000	1.00000	1.00000
AVGU < WSUMU	0.00000	0.00000	0.00000
AVGU ≠ WSUMU	0.00000	0.00000	0.00000

Table D.4: Sign test proportions in noisy situation

	group-change	fragment	hetero-to-homo
MCDP-I > MCDP-II	1.0000	0.6400	0.9700
MCDP-I < MCDP-II	0.0000	0.3600	0.0300
MCDP-I ≠ MCDP-II	1.0000	1.0000	1.0000
MCDP-I > AVG	0.0000	0.8200	0.8300
MCDP-I < AVG	1.0000	0.1800	0.1700
MCDP-I ≠ AVG	1.0000	1.0000	1.0000
MCDP-I > MAX	0.0000	0.6400	0.9900
MCDP-I < MAX	1.0000	0.3600	0.0100
MCDP-I ≠ MAX	1.0000	1.0000	1.0000
MCDP-I > AVGU	0.5800	1.0000	0.9900
MCDP-I < AVGU	0.4200	0.0000	0.0100
MCDP-I ≠ AVGU	1.0000	1.0000	1.0000
MCDP-I > WSUMU	0.1000	1.0000	0.9500
MCDP-I < WSUMU	0.9000	0.0000	0.0500
MCDP-I ≠ WSUMU	1.0000	1.0000	1.0000
MCDP-II > AVG	0.0000	0.7400	0.1100
MCDP-II < AVG	1.0000	0.2600	0.8900
MCDP-II ≠ AVG	1.0000	1.0000	1.0000
MCDP-II > MAX	0.0000	0.5600	0.2900
MCDP-II < MAX	1.0000	0.4400	0.7100
MCDP-II ≠ MAX	1.0000	1.0000	1.0000
MCDP-II > AVGU	0.0000	1.0000	0.2600
MCDP-II < AVGU	1.0000	0.0000	0.7400
MCDP-II ≠ AVGU	1.0000	1.0000	1.0000
MCDP-II > WSUMU	0.0000	1.0000	0.1900
MCDP-II < WSUMU	1.0000	0.0000	0.8100
MCDP-II ≠ WSUMU	1.0000	1.0000	1.0000
AVG > MAX	0.4200	0.0800	1.0000
AVG < MAX	0.5800	0.9200	0.0000
AVG ≠ MAX	1.0000	1.0000	1.0000
AVG > AVGU	1.0000	1.0000	0.9300
AVG < AVGU	0.0000	0.0000	0.0700
AVG ≠ AVGU	1.0000	1.0000	1.0000
AVG > WSUMU	1.0000	0.9900	0.8300
AVG < WSUMU	0.0000	0.0100	0.1700
AVG ≠ WSUMU	1.0000	1.0000	1.0000
MAX > AVGU	1.0000	1.0000	0.3800
MAX < AVGU	0.0000	0.0000	0.6200
MAX ≠ AVGU	1.0000	1.0000	1.0000
MAX > WSUMU	1.0000	1.0000	0.2100
MAX < WSUMU	0.0000	0.0000	0.7900
MAX ≠ WSUMU	1.0000	1.0000	1.0000
AVGU > WSUMU	0.0000	0.0000	0.0400
AVGU < WSUMU	1.0000	1.0000	0.9600
AVGU ≠ WSUMU	1.0000	1.0000	1.0000

TABLE D.5: sign test p-values in complex situation

	group-change	fragment	hetero-to-homo
MCDP-I > MCDP-II	0.00000	0.00000	1.00000
MCDP-I < MCDP-II	1.00000	1.00000	0.00000
MCDP-I ≠ MCDP-II	0.00000	0.00000	0.00000
MCDP-I > AVG	1.00000	0.00000	0.69146
MCDP-I < AVG	0.00000	1.00000	0.38209
MCDP-I ≠ AVG	0.00000	0.00000	0.76418
MCDP-I > MAX	1.00000	0.00000	0.00000
MCDP-I < MAX	0.00000	1.00000	1.00000
MCDP-I ≠ MAX	0.00000	0.00000	0.00000
MCDP-I > AVGU	1.00000	0.00000	0.00000
MCDP-I < AVGU	0.00000	1.00000	1.00000
MCDP-I ≠ AVGU	0.00000	0.00000	0.00000
MCDP-I > WSUMU	1.00000	0.00000	0.00000
MCDP-I < WSUMU	0.00000	1.00000	1.00000
MCDP-I ≠ WSUMU	0.00000	0.00000	0.00000
MCDP-II > AVG	1.00000	1.00000	0.00000
MCDP-II < AVG	0.00000	0.00000	1.00000
MCDP-II ≠ AVG	0.00000	0.00000	0.00000
MCDP-II > MAX	1.00000	1.00000	0.00000
MCDP-II < MAX	0.00000	0.00000	1.00000
MCDP-II ≠ MAX	0.00000	0.00000	0.00000
MCDP-II > AVGU	1.00000	0.00000	0.00000
MCDP-II < AVGU	0.00000	1.00000	1.00000
MCDP-II ≠ AVGU	0.00000	0.00000	0.00000
MCDP-II > WSUMU	1.00000	0.00000	0.00000
MCDP-II < WSUMU	0.00000	1.00000	1.00000
MCDP-II ≠ WSUMU	0.00000	0.00000	0.00000
AVG > MAX	1.00000	1.00000	0.00000
AVG < MAX	0.00000	0.00000	1.00000
AVG ≠ MAX	0.00000	0.00000	0.00000
AVG > AVGU	0.00000	0.00000	0.00000
AVG < AVGU	1.00000	1.00000	1.00000
AVG ≠ AVGU	0.00000	0.00000	0.00000
AVG > WSUMU	0.00000	0.00000	0.00000
AVG < WSUMU	1.00000	1.00000	1.00000
AVG ≠ WSUMU	0.00000	0.00000	0.00000
MAX > AVGU	0.00000	0.00000	0.00621
MAX < AVGU	1.00000	1.00000	0.99653
MAX ≠ AVGU	0.00000	0.00000	0.01242
MAX > WSUMU	0.00000	0.00000	0.00347
MAX < WSUMU	1.00000	1.00000	0.99813
MAX ≠ WSUMU	0.00000	0.00000	0.00693
AVGU > WSUMU	1.00000	1.00000	0.78924
AVGU < WSUMU	0.00000	0.00000	0.27336
AVGU ≠ WSUMU	0.00000	0.00000	0.54671

TABLE D.6: Sign test proportions in complex situation

	groupchange	fragment	heterotohomo
MCDP-I > MCDP-II	1.0000	1.0000	0.0000
MCDP-I < MCDP-II	0.0000	0.0000	1.0000
MCDP-I ≠ MCDP-II	1.0000	1.0000	1.0000
MCDP-I > AVG	0.0000	0.8500	0.4800
MCDP-I < AVG	1.0000	0.1500	0.5200
MCDP-I ≠ AVG	1.0000	1.0000	1.0000
MCDP-I > MAX	0.0000	0.8300	0.7900
MCDP-I < MAX	1.0000	0.1700	0.2100
MCDP-I ≠ MAX	1.0000	1.0000	1.0000
MCDP-I > AVGU	0.0200	0.9800	0.8700
MCDP-I < AVGU	0.9800	0.0200	0.1300
MCDP-I ≠ AVGU	1.0000	1.0000	1.0000
MCDP-I > WSUMU	0.0100	0.9800	0.8900
MCDP-I < WSUMU	0.9900	0.0200	0.1100
MCDP-I ≠ WSUMU	1.0000	1.0000	1.0000
MCDP-II > AVG	0.0000	0.2600	1.0000
MCDP-II < AVG	1.0000	0.7400	0.0000
MCDP-II ≠ AVG	1.0000	1.0000	1.0000
MCDP-II > MAX	0.0000	0.2100	1.0000
MCDP-II < MAX	1.0000	0.7900	0.0000
MCDP-II ≠ MAX	1.0000	1.0000	1.0000
MCDP-II > AVGU	0.0000	0.8700	1.0000
MCDP-II < AVGU	1.0000	0.1300	0.0000
MCDP-II ≠ AVGU	1.0000	1.0000	1.0000
MCDP-II > WSUMU	0.0000	0.8300	1.0000
MCDP-II < WSUMU	1.0000	0.1700	0.0000
MCDP-II ≠ WSUMU	1.0000	1.0000	1.0000
AVG > MAX	0.1900	0.2000	1.0000
AVG < MAX	0.8100	0.8000	0.0000
AVG ≠ MAX	1.0000	1.0000	1.0000
AVG > AVGU	1.0000	0.9800	0.9200
AVG < AVGU	0.0000	0.0200	0.0800
AVG ≠ AVGU	1.0000	1.0000	1.0000
AVG > WSUMU	1.0000	0.9800	0.9000
AVG < WSUMU	0.0000	0.0200	0.1000
AVG ≠ WSUMU	1.0000	1.0000	1.0000
MAX > AVGU	1.0000	0.9800	0.6300
MAX < AVGU	0.0000	0.0200	0.3700
MAX ≠ AVGU	1.0000	1.0000	1.0000
MAX > WSUMU	1.0000	0.9800	0.6400
MAX < WSUMU	0.0000	0.0200	0.3600
MAX ≠ WSUMU	1.0000	1.0000	1.0000
AVGU > WSUMU	0.0000	0.0300	0.4600
AVGU < WSUMU	1.0000	0.9700	0.5300
AVGU ≠ WSUMU	1.0000	1.0000	0.9900

TABLE D.7: Sign test p-values for mixed change scenario

	All	Two
MCDP-I > MCDP-II	1.00000	1.00000
MCDP-I < MCDP-II	0.00000	0.00000
MCDP-I ≠ MCDP-II	0.00000	0.00000
MCDP-I > AVG	0.46017	0.00000
MCDP-I < AVG	0.61791	1.00000
MCDP-I ≠ AVG	0.92034	0.00000
MCDP-I > MAX	0.00000	0.00000
MCDP-I < MAX	1.00000	1.00000
MCDP-I ≠ MAX	0.00000	0.00000
MCDP-I > AVGU	0.00000	0.00000
MCDP-I < AVGU	1.00000	1.00000
MCDP-I ≠ AVGU	0.00000	0.00000
MCDP-I > WSUMU	0.00000	0.00000
MCDP-I < WSUMU	1.00000	1.00000
MCDP-I ≠ WSUMU	0.00000	0.00000
MCDP-II > AVG	0.00000	0.00000
MCDP-II < AVG	1.00000	1.00000
MCDP-II ≠ AVG	0.00000	0.00000
MCDP-II > MAX	0.00000	0.00000
MCDP-II < MAX	1.00000	1.00000
MCDP-II ≠ MAX	0.00000	0.00000
MCDP-II > AVGU	0.00000	0.00000
MCDP-II < AVGU	1.00000	1.00000
MCDP-II ≠ AVGU	0.00000	0.00000
MCDP-II > WSUMU	0.00000	0.00000
MCDP-II < WSUMU	1.00000	1.00000
MCDP-II ≠ WSUMU	0.00000	0.00000
AVG > MAX	0.00000	0.99903
AVG < MAX	1.00000	0.00187
AVG ≠ MAX	0.00000	0.00373
AVG > AVGU	0.00000	0.00002
AVG < AVGU	1.00000	0.99999
AVG ≠ AVGU	0.00000	0.00004
AVG > WSUMU	0.00000	0.00000
AVG < WSUMU	1.00000	1.00000
AVG ≠ WSUMU	0.00000	0.00001
MAX > AVGU	0.00002	0.00011
MAX < AVGU	0.99999	0.99995
MAX ≠ AVGU	0.00004	0.00022
MAX > WSUMU	0.00097	0.00002
MAX < WSUMU	0.99952	0.99999
MAX ≠ WSUMU	0.00194	0.00004
AVGU > WSUMU	0.97128	0.01786
AVGU < WSUMU	0.04457	0.98928
AVGU ≠ WSUMU	0.08913	0.03573

TABLE D.8: Sign test proportions for mixed change scenario in complex situation

	All	Two
MCDP-I > MCDP-II	0.0000	0.0000
MCDP-I < MCDP-II	1.0000	1.0000
MCDP-I ≠ MCDP-II	1.0000	1.0000
MCDP-I > AVG	0.5100	0.8700
MCDP-I < AVG	0.4900	0.1300
MCDP-I ≠ AVG	1.0000	1.0000
MCDP-I > MAX	0.9400	0.7500
MCDP-I < MAX	0.0600	0.2500
MCDP-I ≠ MAX	1.0000	1.0000
MCDP-I > AVGU	0.9800	0.9800
MCDP-I < AVGU	0.0200	0.0200
MCDP-I ≠ AVGU	1.0000	1.0000
MCDP-I > WSUMU	0.9700	0.9600
MCDP-I < WSUMU	0.0300	0.0400
MCDP-I ≠ WSUMU	1.0000	1.0000
MCDP-II > AVG	1.0000	1.0000
MCDP-II < AVG	0.0000	0.0000
MCDP-II ≠ AVG	1.0000	1.0000
MCDP-II > MAX	1.0000	1.0000
MCDP-II < MAX	0.0000	0.0000
MCDP-II ≠ MAX	1.0000	1.0000
MCDP-II > AVGU	1.0000	1.0000
MCDP-II < AVGU	0.0000	0.0000
MCDP-II ≠ AVGU	1.0000	1.0000
MCDP-II > WSUMU	1.0000	1.0000
MCDP-II < WSUMU	0.0000	0.0000
MCDP-II ≠ WSUMU	1.0000	1.0000
AVG > MAX	0.9900	0.3500
AVG < MAX	0.0100	0.6500
AVG ≠ MAX	1.0000	1.0000
AVG > AVGU	0.9700	0.7100
AVG < AVGU	0.0300	0.2900
AVG ≠ AVGU	1.0000	1.0000
AVG > WSUMU	0.9600	0.7300
AVG < WSUMU	0.0400	0.2700
AVG ≠ WSUMU	1.0000	1.0000
MAX > AVGU	0.7100	0.6900
MAX < AVGU	0.2900	0.3100
MAX ≠ AVGU	1.0000	1.0000
MAX > WSUMU	0.6600	0.7100
MAX < WSUMU	0.3400	0.2900
MAX ≠ WSUMU	1.0000	1.0000
AVGU > WSUMU	0.4100	0.6100
AVGU < WSUMU	0.5900	0.3900
AVGU ≠ WSUMU	1.0000	1.0000

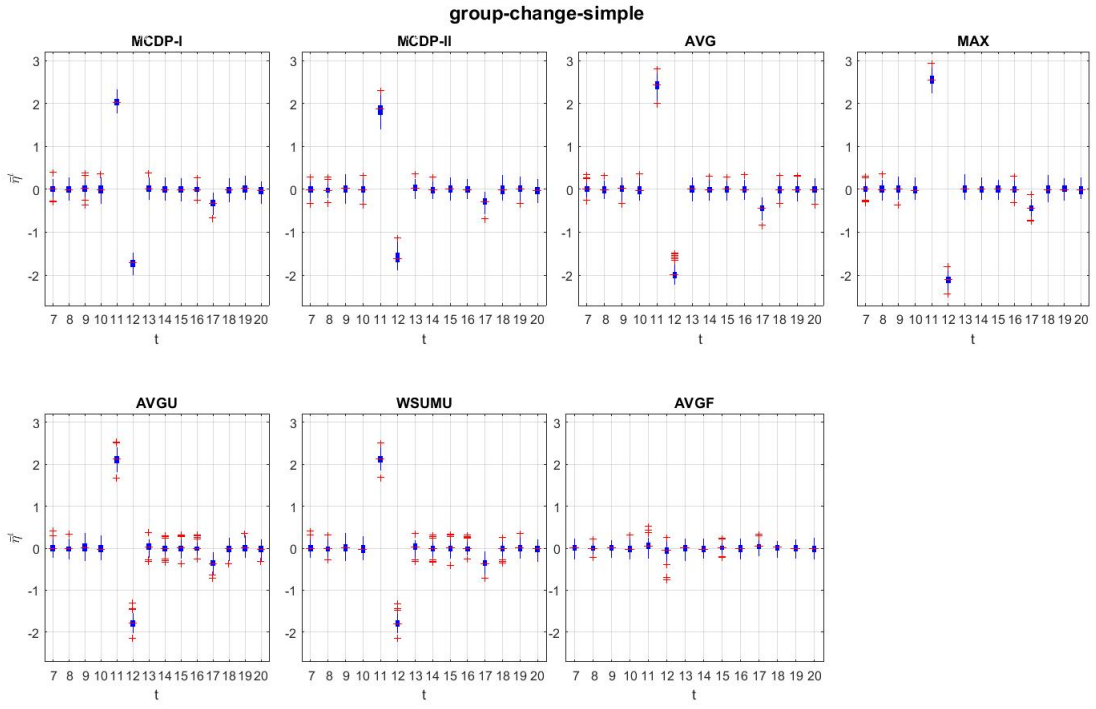


FIGURE D.14: **Observing $\bar{\boldsymbol{\eta}}^t$ on group-change in simple situation using $w = 5$. The change-point is at $t = 11$.** Except for AVGF, all methods show good detection performance at $t = 11$.

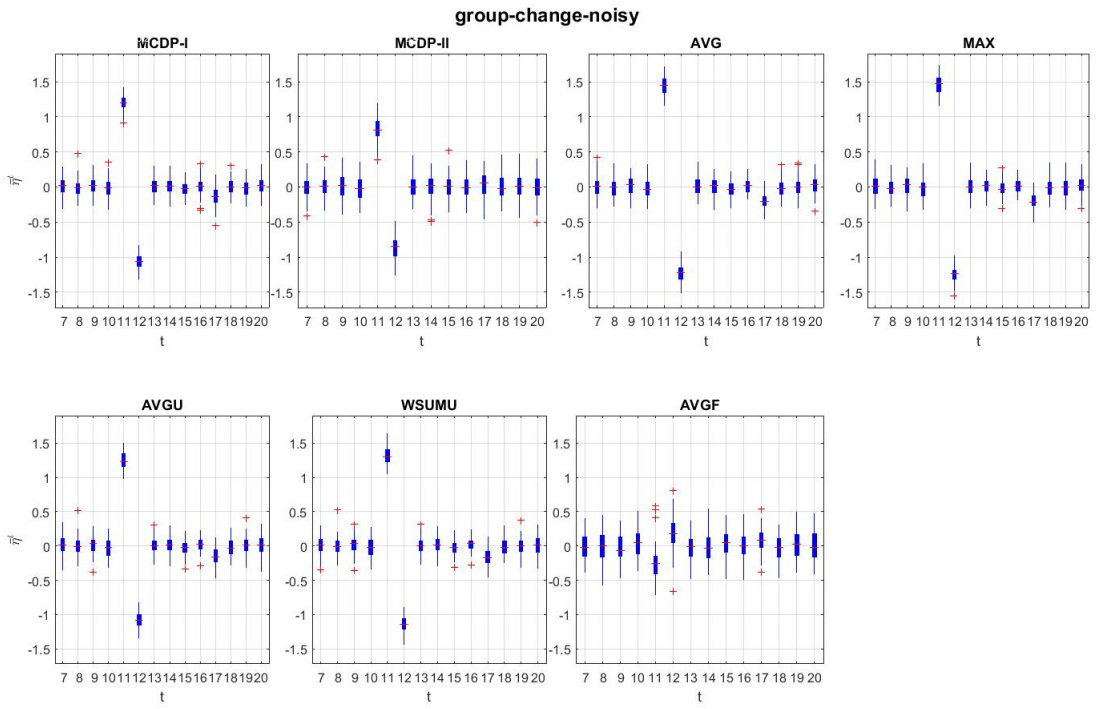


FIGURE D.15: **Observing $\bar{\boldsymbol{\eta}}^t$ on group-change in noisy situation using $w = 5$. The change-point is at $t = 11$.** AVGF does not detect the change at $t = 11$. All other methods show good detection performance at $t = 11$.

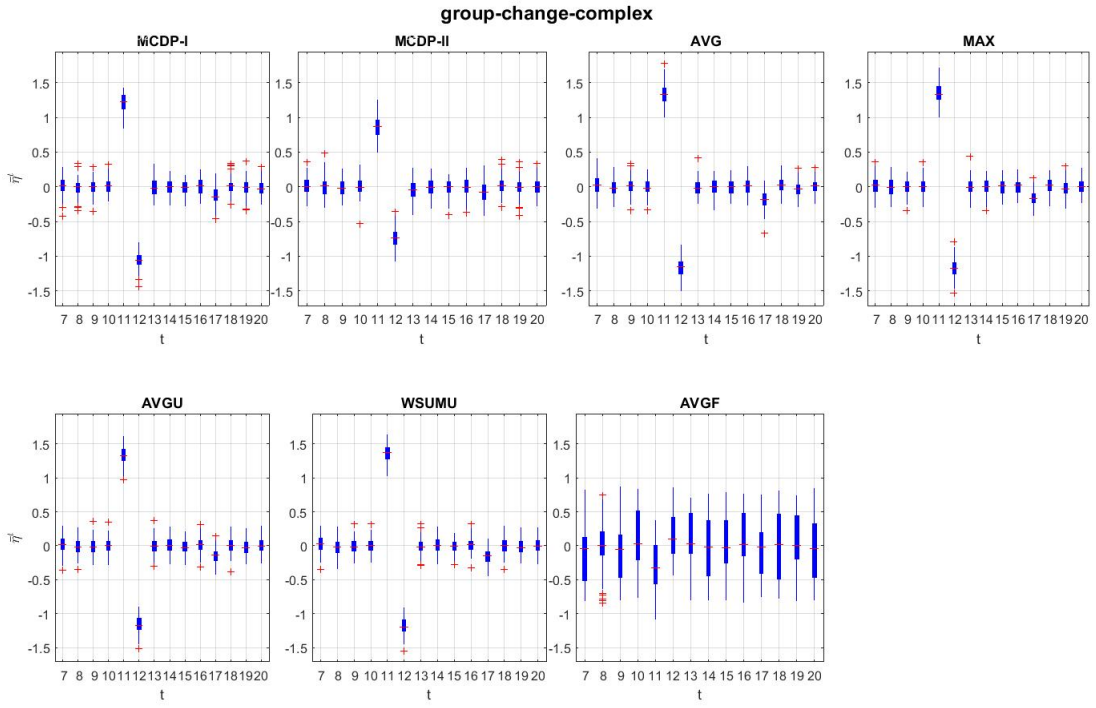


FIGURE D.16: **Observing $\bar{\eta}^t$ on group-change in complex situation using $w = 5$. The change-point is at $t = 11$.** AVGF does not detect the change at $t = 11$. All other methods show good detection performance at $t = 11$.

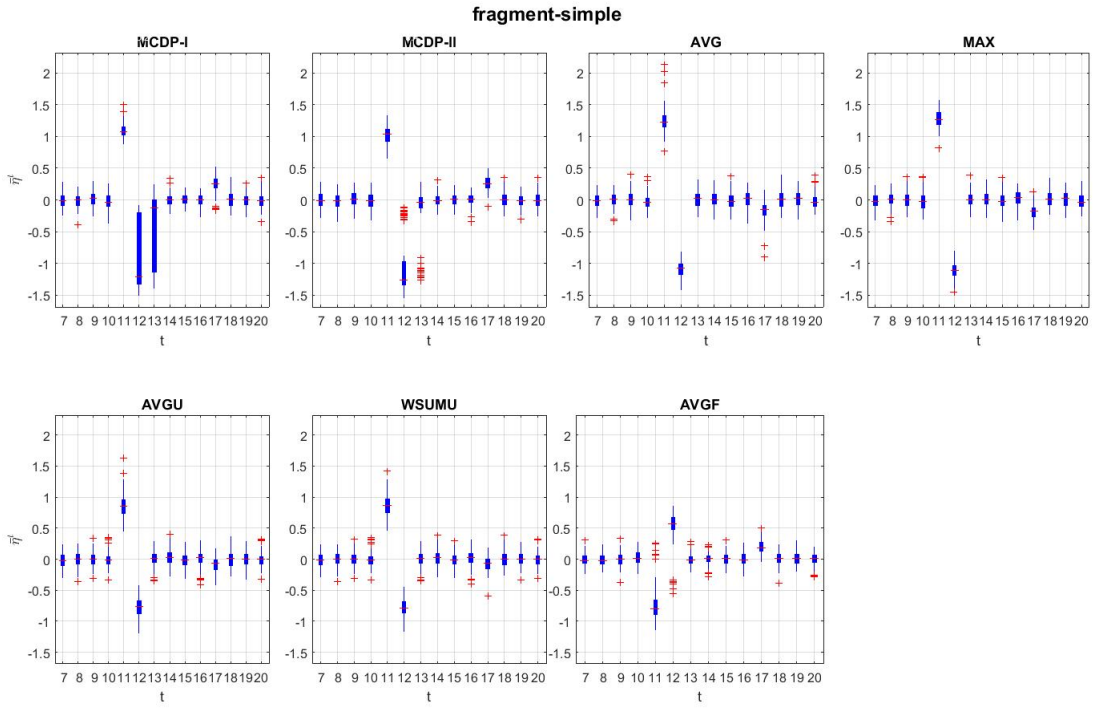


FIGURE D.17: **Observing $\bar{\boldsymbol{\eta}}^t$ on fragment in simple situation using $w = 5$. The change-point is at $t = 11$.** AVGF does not detect the change at $t = 11$. All other methods show good detection performance at $t = 11$.

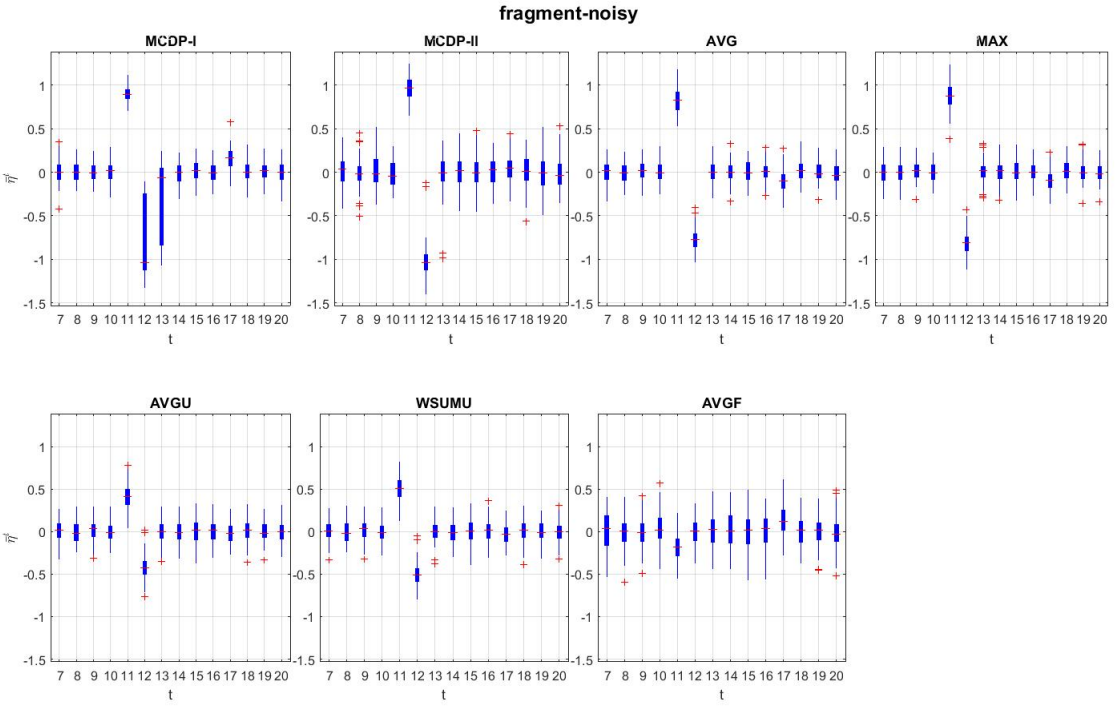


FIGURE D.18: **Observing $\bar{\boldsymbol{\eta}}^t$ on fragment in noisy situation using $w = 5$. The change-point is at $t = 11$.** AVGF does not detect the change at $t = 11$. All other methods show good detection performance at $t = 11$.

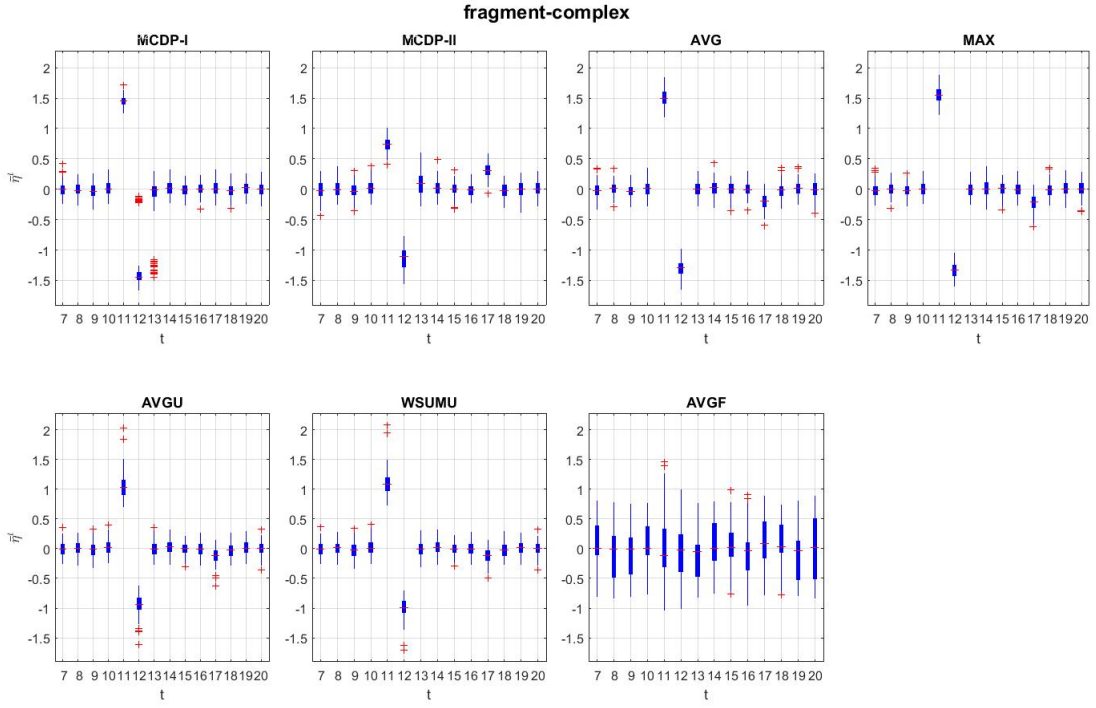


FIGURE D.19: **Observing $\bar{\boldsymbol{\eta}}^t$ on fragment in complex situation using $w = 5$. The change-point is at $t = 11$.** AVGF does not detect the change at $t = 11$. All other methods show good detection performance at $t = 11$.

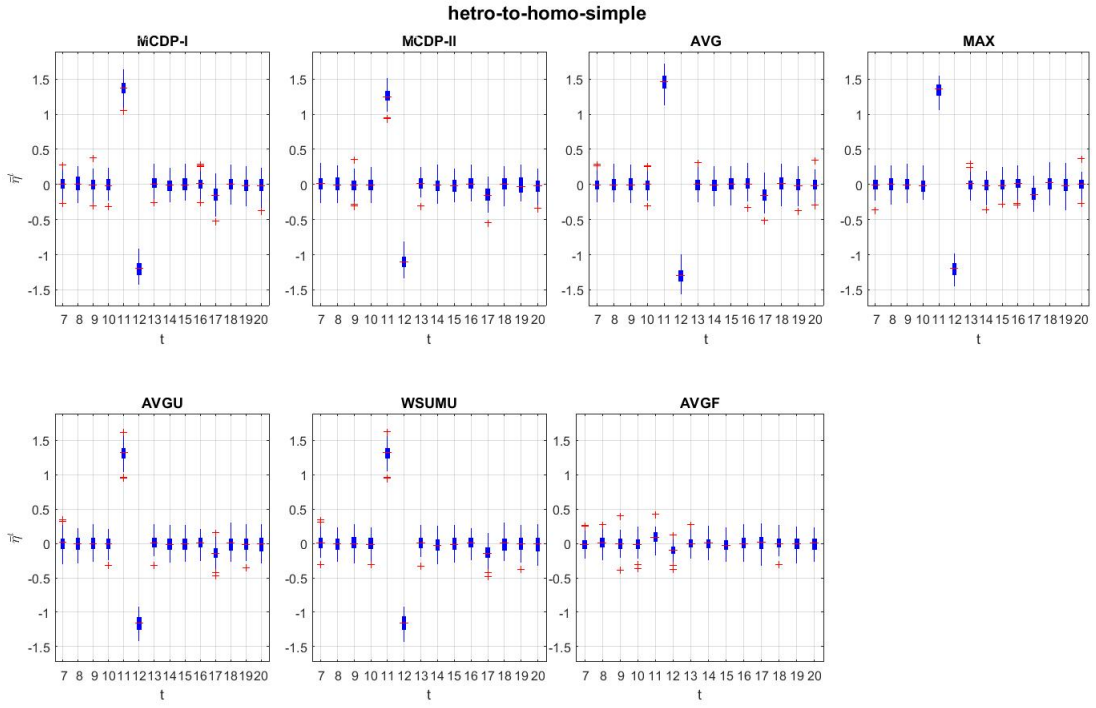


FIGURE D.20: **Observing $\bar{\boldsymbol{\eta}}^t$ on hetero-to-homo in simple situation using $w = 5$. The change-point is at $t = 11$.** All methods show a detection at $t = 11$, but the increase of $\bar{\boldsymbol{\eta}}^{11}_{AVGF}$ is comparatively very small.

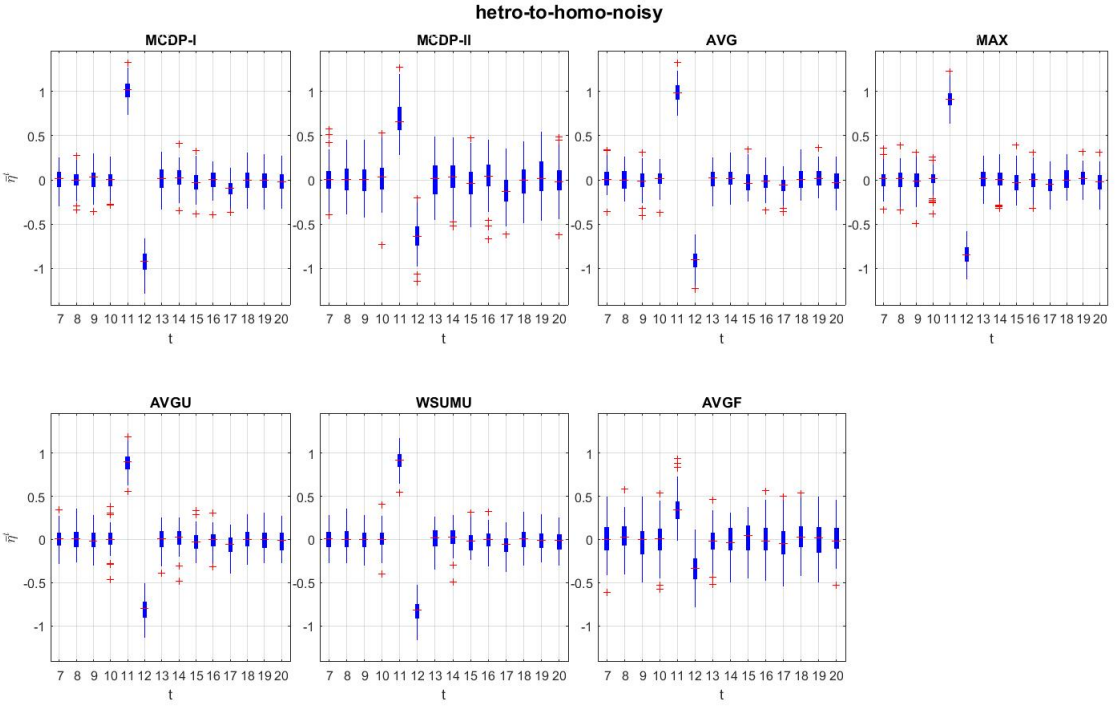


FIGURE D.21: **Observing $\bar{\boldsymbol{\eta}}^t$ on hetero-to-homo in noisy situation using $w = 5$. The change-point is at $t = 11$.** All methods show a detection at $t = 11$.

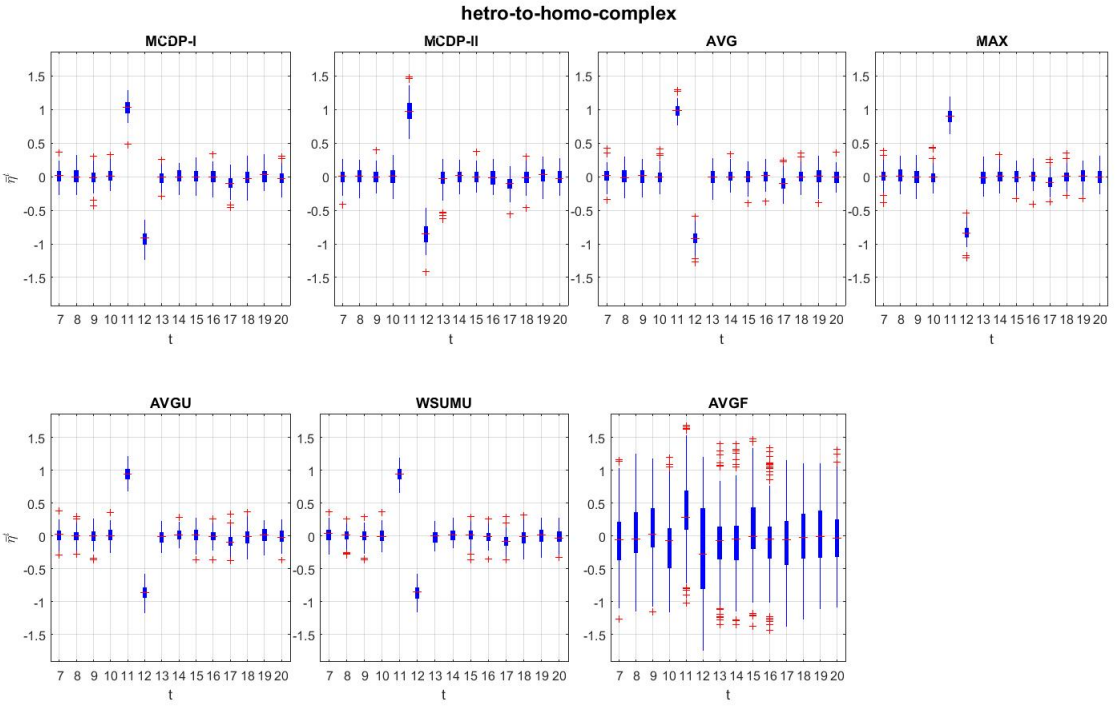


FIGURE D.22: **Observing $\bar{\boldsymbol{\eta}}^t$ on hetero-to-homo in complex situation using $w = 5$. The change-point is at $t = 11$.** All methods show a detection at $t = 11$. However, $\bar{\boldsymbol{\eta}}^t_{AVGF}$'s are wide and contain a large number of outliers.

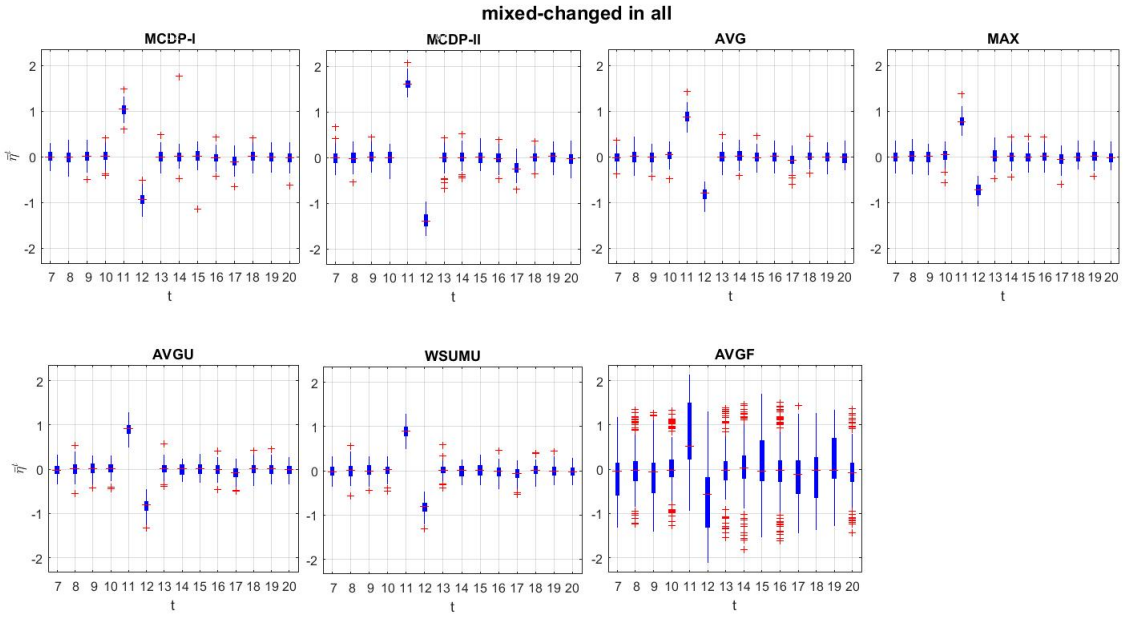


FIGURE D.23: **Observing $\bar{\boldsymbol{\eta}}^t$ on mixed-all using $w = 5$. The change-point is at $t = 11$.** For all methods, there is an increase for $\bar{\boldsymbol{\eta}}^{11}$. However, $\bar{\boldsymbol{\eta}}^{11}_{AVGF}$ is wide and extends below zero.

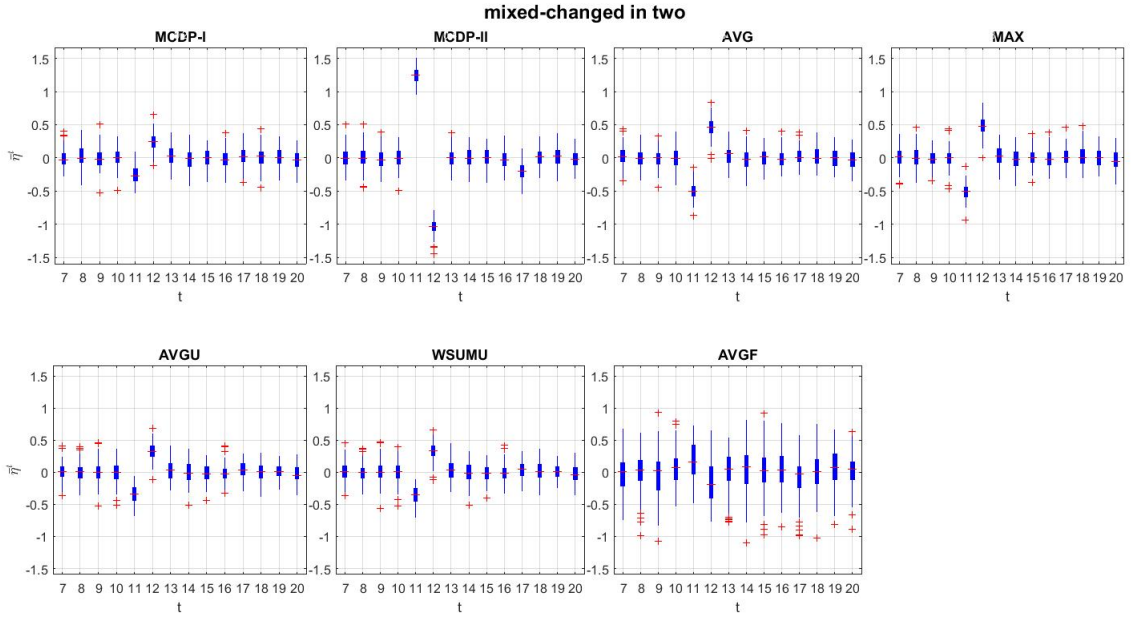


FIGURE D.24: **Observing $\bar{\boldsymbol{\eta}}^t$ on mixed-two using $w = 5$. The change-point is at $t = 11$.** Only MCDP-II and AVGF indicate an increase for $\bar{\boldsymbol{\eta}}^{11}$, while MCDP-II showing a clearer detection.

Bibliography

Achlioptas, D. and F. McSherry (2007). Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM) 54*(2), 9.

Aicher, C., A. Z. Jacobs, and A. Clauset (2014). Learning latent block structure in weighted networks. *arXiv preprint arXiv:1404.0431*.

Akoglu, L. and C. Faloutsos (2010). Event detection in time series of mobile communication graphs. In *Army Science Conference*, pp. 77–79.

Akoglu, L. and C. Faloutsos (2013). Anomaly, event, and fraud detection in large network datasets. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 773–774. ACM.

Amini, A. A., A. Chen, P. J. Bickel, and E. Levina (2013, 08). Pseudo-likelihood methods for community detection in large sparse networks. *Ann. Statist. 41*(4), 2097–2122.

Amini, A. A., A. Chen, P. J. Bickel, E. Levina, et al. (2013). Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics 41*(4), 2097–2122.

Anderson, G. W., A. Guionnet, and O. Zeitouni (2010). *An introduction to random matrices*, Volume 118. Cambridge university press.

Bacry, E., S. Gaïffas, and J.-F. Muzy (2015). A generalization error bound for sparse and low-rank multivariate hawkes processes. *arXiv preprint arXiv:1501.00725*.

Banfield, J. D. and A. E. Raftery (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, 803–821.

Belabbas, M.-A. and P. J. Wolfe (2009). Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences 106*(2), 369–374.

Belkin, M. and P. Niyogi (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation 15*(6), 1373–1396.

Bickel, P. J. and A. Chen (2009). A nonparametric view of network models and newman–girvan and other modularities. *Proceedings of the National Academy of Sciences 106*(50), 21068–21073.

Borg, I. and P. J. Groenen (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Cai, D., Z. Shao, X. He, X. Yan, and J. Han (2005). Community mining from multi-relational networks. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 445–452. Springer.

Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate behavioral research 1*(2), 245–276.

Chandola, V., A. Banerjee, and V. Kumar (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR) 41*(3), 15.

Chen, Z., W. Hendrix, and N. F. Samatova (2012). Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems 39*(1), 59–85.

Cheng, H., P.-N. Tan, C. Potter, and S. Klooster (2008). A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series. In *2008 IEEE International Conference on Data Mining Workshops*, pp. 349–358. IEEE.

Chung, F. R. (1997). *Spectral graph theory*, Volume 92. American Mathematical Soc.

Clauset, A., C. R. Shalizi, and M. E. Newman (2009). Power-law distributions in empirical data. *SIAM review 51*(4), 661–703.

Cooley, W. W. and P. R. Lohnes (1971). *Multivariate data analysis*. J. Wiley.

Cootes, T. F., C. J. Taylor, et al. (2004). Statistical models of appearance for computer vision.

Cootes, T. F., C. J. Taylor, D. H. Cooper, and J. Graham (1992). Training models of shape from sets of examples. In *BMVC92*, pp. 9–18. Springer.

Crosilla, F. and A. Beinat (2002). Use of generalised procrustes analysis for the photogrammetric block adjustment by independent models. *ISPRS Journal of Photogrammetry and Remote Sensing 56*(3), 195–209.

De Lathauwer, L., B. De Moor, and J. Vandewalle (2000a). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications 21*(4), 1253–1278.

De Lathauwer, L., B. De Moor, and J. Vandewalle (2000b). On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications 21*(4), 1324–1342.

Dryden, I. L. and K. V. Mardia (1998). *Statistical shape analysis*, Volume 4. Wiley Chichester.

Durante, D. and D. B. Dunson (2014). Bayesian dynamic financial networks with time-varying predictors. *Statistics & Probability Letters 93*, 19–26.

Eckart, C. and G. Young (1936). The approximation of one matrix by another of lower rank. *Psychometrika 1*(3), 211–218.

Erdős, P. and A. Rényi (1959). On random graphs. *Publicationes Mathematicae Debrecen 6*, 290–297.

Fanaee-T, H. and J. Gama (2016). Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems 98*, 130–147.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery in databases. *AI magazine 17*(3), 37.

Ghosh, P., P. Gill, S. Muthukumarana, and T. Swartz (2010). A semiparametric bayesian approach to network modelling using dirichlet process prior distributions. *Australian & New Zealand Journal of Statistics 52*(3), 289–302.

Goldenberg, A., A. X. Zheng, S. E. Fienberg, and E. M. Airoldi (2010). A survey of statistical network models. *Foundations and Trends® in Machine Learning 2*(2), 129–233.

Greene, D. and P. Cunningham (2013). Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th annual ACM web science conference*, pp. 118–121. ACM.

Gruen, A. W. and M. D. Akca (2003). Generalized procrustes analysis and its applications in photogrammetry.

Guttman, L. (1954). Some necessary conditions for common-factor analysis. *Psychometrika 19*(2), 149–161.

Han, Q., K. Xu, and E. Airoldi (2015). Consistent estimation of dynamic and multi-layer block models. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1511–1520.

Handcock, M. S., A. E. Raftery, and J. M. Tantrum (2007). Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society) 170*(2), 301–354.

Harshman, R. A. (1970). Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis.

Harshman, R. A. (1972). Determination and proof of minimum uniqueness conditions for parafac1. *UCLA Working Papers in phonetics 22*(111-117), 3.

Hawkins, D. M. (1980). *Identification of outliers*, Volume 11. Springer.

Heard, N. A., D. J. Weston, K. Platanioti, D. J. Hand, et al. (2010). Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics 4*(2), 645–662.

Hirose, S., K. Yamanishi, T. Nakata, and R. Fujimaki (2009). Network anomaly detection based on eigen equation compression. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1185–1194. ACM.

Hoff, P. D., A. E. Raftery, and M. S. Handcock (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association 97*(460), 1090–1098.

Holland, P. W., K. B. Laskey, and S. Leinhardt (1983). Stochastic blockmodels: First steps. *Social networks 5*(2), 109–137.

Huang, Y. and X. Gao (2014). Clustering on heterogeneous networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4*(3), 213–233.

Idé, T. and H. Kashima (2004). Eigenspace-based anomaly detection in computer systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 440–449. ACM.

Idé, T., A. C. Lozano, N. Abe, and Y. Liu (2009). Proximity-based anomaly detection using sparse structure learning. In *SDM*, pp. 97–108. SIAM.

Jackson, D. A. (1993). Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches. *Ecology*, 2204–2214.

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Jolliffe, I. T. (1972). Discarding variables in a principal component analysis. i: Artificial data. *Applied statistics*, 160–173.

Joseph, A. and B. Yu (2013). Impact of regularization on spectral clustering. *arXiv preprint arXiv:1312.1733*.

Karrer, B. and M. E. Newman (2011). Stochastic blockmodels and community structure in networks. *Physical Review E 83*(1), 016107.

Khoa, N. L. D. and S. Chawla (2012). Large scale spectral clustering using resistance distance and spielman-teng solvers. In *Discovery Science*, pp. 7–21. Springer.

Klimt, B. and Y. Yang (2004). Introducing the enron corpus. In *CEAS*.

Kolaczyk, E. D. (2009). *Statistical analysis of network data: methods and models*. Springer.

Kolda, T. G. (2006). *Multilinear operators for higher-order decompositions*. United States. Department of Energy.

Kolda, T. G. and B. W. Bader (2009). Tensor decompositions and applications. *SIAM review 51*(3), 455–500.

Kolda, T. G. and J. Sun (2008). Scalable tensor decompositions for multi-aspect data mining. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 363–372. IEEE.

Koujaku, S., M. Kudo, I. Takigawa, and H. Imai (2013). Structrual change point detection for evolutional networks. In *Proceedings of the World Congress on Engineering*, Volume 1.

Koujaku, S., M. Kudo, I. Takigawa, and H. Imai (2015). Community change detection in dynamic networks in noisy environment. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 793–798. International World Wide Web Conferences Steering Committee.

Koutra, D., E. E. Papalexakis, and C. Faloutsos (2012). Tensorsplat: Spotting latent anomalies in time. In *Informatics (PCI), 2012 16th Panhellenic Conference on*, pp. 144–149. IEEE.

Lambert, Z. V., A. R. Wildt, and R. M. Durand (1990). Assessing sampling variation relative to number-of-factors criteria. *Educational and Psychological Measurement 50*(1), 33–48.

Le, C. M., E. Levina, and R. Vershynin (2015). Sparse random graphs: regularization and concentration of the laplacian. *arXiv preprint arXiv:1502.03049*.

Le, C. M. and R. Vershynin (2015). Concentration and regularization of random graphs. *arXiv preprint arXiv:1506.00669*.

Liu, X., S. Ji, W. Glänzel, and B. De Moor (2013). Multiview partitioning via tensor methods. *IEEE Transactions on Knowledge and Data Engineering 25*(5), 1056–1069.

Lovász, L. (1993). Random walks on graphs: A survey. *Combinatorics: Paul Erdös is 80*, 353–397.

Marchette, D. (2012). Scan statistics on graphs. *Wiley Interdisciplinary Reviews: Computational Statistics 4*(5), 466–473.

Martin, T., X. Zhang, and M. Newman (2014). Localization and centrality in networks. *Physical Review E 90*(5), 052808.

McPherson, M., L. Smith-Lovin, and J. M. Cook (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 415–444.

Meilă, M. (2003). Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pp. 173–187. Springer.

Mihail, M. and C. Papadimitriou (2002). On the eigenvalue power law. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pp. 254–262. Springer.

Neil, J., C. Hash, A. Brugh, M. Fisk, and C. B. Storlie (2013). Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics 55*(4), 403–414.

Neil, J., B. Uphoff, C. Hash, and C. Storlie (2013). Towards improved detection of attackers in computer networks: New edges, fast updating, and host agents. In *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*, pp. 218–224. IEEE.

Newman, M. E. (2004a). Analysis of weighted networks. *Physical Review E 70*(5), 056131.

Newman, M. E. (2004b). Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems 38*(2), 321–330.

Ng, A. Y., M. I. Jordan, and Y. Weiss (2001). On spectral clustering1 analysis and an algorithm. *Proceedings of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press 14*, 849–856.

Nickel, C. L. M. (2007). *Random dot product graphs: A model for social networks*, Volume 68.

Nickel, M. (2013). *Tensor factorization for relational learning*. Ph. D. thesis, lmu.

Oliveira, R. I. (2009). Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint arXiv:0911.0600*.

Pandit, S., D. H. Chau, S. Wang, and C. Faloutsos (2007). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pp. 201–210. ACM.

Papalexakis, E. E., L. Akoglu, and D. Ience (2013). Do more views of a graph help? community detection and clustering in multi-graphs. In *Information Fusion (FUSION), 2013 16th International Conference on*, pp. 899–905. IEEE.

Papalexakis, E. E., C. Faloutsos, and N. D. Sidiropoulos (2012). Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pp. 521–536. Springer.

Peel, L. and A. Clauset (2015). Detecting change points in the large-scale structure of evolving networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Poole, D. (2014). *Linear algebra: A modern introduction*. Cengage Learning.

Priebe, C. E., J. M. Conroy, D. J. Marchette, and Y. Park (2005). Scan statistics on enron graphs. *Computational & Mathematical Organization Theory 11*(3), 229–247.

Qin, T. and K. Rohe (2013). Regularized spectral clustering under the degree-corrected stochastic blockmodel. In *Advances in Neural Information Processing Systems*, pp. 3120–3128.

Ranshous, S., S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova (2015). Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics 7*(3), 223–247.

Rohe, K., S. Chatterjee, and B. Yu (2011). Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 1878–1915.

Rohe, K., T. Qin, and B. Yu (2015). Co-clustering for directed graphs: the stochastic co-blockmodel and spectral algorithm di-sim. *arXiv preprint arXiv:1204.2296*.

Ross, A. (2004). Procrustes analysis. *Course report, Department of Computer Science and Engineering, University of South Carolina*.

Rossi, R. A., B. Gallagher, J. Neville, and K. Henderson (2013). Modeling dynamic behavior in large evolving graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 667–676. ACM.

Rougier, C., J. Meunier, A. St-Arnaud, and J. Rousseau (2008). Procrustes shape analysis for fall detection. In *The Eighth International Workshop on Visual Surveillance-VS2008*.

Saerens, M., F. Fouss, L. Yen, and P. Dupont (2004). The principal components analysis of a graph, and its relationships to spectral clustering. In *European Conference on Machine Learning*, pp. 371–383. Springer.

Salter-Townshend, M., A. White, I. Gollini, and T. B. Murphy (2012). Review of statistical network analysis: models, algorithms, and software. *Statistical Analysis and Data Mining 5*(4), 243–264.

Savage, D., X. Zhang, X. Yu, P. Chou, and Q. Wang (2014). Anomaly detection in online social networks. *Social Networks 39*, 62–70.

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika 31*(1), 1–10.

Sengupta, S. and Y. Chen (2015). Spectral clustering in heterogeneous networks. *Statistica Sinica:Vol. 25, No. 3, 1081-1106*.

Shi, J. and J. Malik (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*(8), 888–905.

Singh, P. and M. Singh (2015). Fraud detection by monitoring customer behavior and activities. *International Journal of Computer Applications 111*(11).

Skillicorn, D. (2007a). *Understanding complex datasets: data mining with matrix decompositions*. CRC press.

Skillicorn, D. B. (2007b). Detecting anomalies in graphs. In *Intelligence and Security Informatics, 2007 IEEE*, pp. 209–216. IEEE.

Snijders, T. A. and K. Nowicki (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification 14*(1), 75–100.

Sricharan, K. and K. Das (2014). Localizing anomalous changes in time-evolving graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1347–1358. ACM.

Stegmann, M. B. and D. D. Gomez (2002). A brief introduction to statistical shape analysis. *Informatics and mathematical modelling, Technical University of Denmark, DTU 15*, 11.

Sun, J., C. Faloutsos, S. Papadimitriou, and P. S. Yu (2007). Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 687–696. ACM.

Sun, J., D. Tao, and C. Faloutsos (2006). Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 374–383. ACM.

Sussman, D. L. (2014). *Foundations of Adjacency Spectral Embedding*. Ph. D. thesis, Ph. D. thesis, Johns Hopkins University.

Sussman, D. L., M. Tang, D. E. Fishkind, and C. E. Priebe (2012). A consistent adjacency spectral embedding for stochastic blockmodel graphs. *Journal of the American Statistical Association 107*(499), 1119–1128.

Sussman, D. L., M. Tang, and C. E. Priebe (2014). Consistent latent position estimation and vertex classification for random dot product graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 36*(1), 48–57.

Suwan, S. (2015). Empirical bayes estimation for random dot product graph representation of the stochastic blockmodel.

Tang, L., H. Liu, J. Zhang, and Z. Nazeri (2008). Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 677–685. ACM.

Tang, L., X. Wang, and H. Liu (2012). Community detection via heterogeneous interaction analysis. *Data Mining and Knowledge Discovery 25*(1), 1–33.

Tang, M., A. Athreya, D. L. Sussman, V. Lyzinski, and C. E. Priebe (2014). Two-sample hypothesis testing for random dot product graphs via adjacency spectral embedding. *arXiv preprint arXiv:1403.7249*.

Tang, W., Z. Lu, and I. S. Dhillon (2009). Clustering with multiple graphs. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 1016–1021. IEEE.

ten Berge, J. M., J. de Leeuw, and P. M. Kroonenberg (1987). Some additional results on principal components analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 52*(2), 183–191.

Tong, H. and C.-Y. Lin (2011). Non-negative residual matrix factorization with application to graph anomaly detection. In *SDM*, pp. 143–153. SIAM.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika 31*(3), 279–311.

Vaswani, N., A. K. Roy-Chowdhury, and R. Chellappa (2005). " shape activity": a continuous-state hmm for moving/deforming shapes with application to abnormal activity detection. *Image Processing, IEEE Transactions on 14*(10), 1603–1616.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing 17*(4), 395–416.

Wagner, D. and F. Wagner (1993). Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pp. 744–750. Springer.

Wang, B., J. M. Phillips, R. Schreiber, D. M. Wilkinson, N. Mishra, and R. Tarjan (2008). Spatial scan statistics for graph clustering. In *SDM*, pp. 727–738. SIAM.

Wang, Y. J. and G. Y. Wong (1987). Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association 82*(397), 8–19.

Watts, D. J. and S. H. Strogatz (1998). Collective dynamics of 'small-world'networks. *nature 393*(6684), 440–442.

Wigner, E. P. (1993). Characteristic vectors of bordered matrices with infinite dimensions i. In *The Collected Works of Eugene Paul Wigner*, pp. 524–540. Springer.

Young, S. J. and E. R. Scheinerman (2007). Random dot product graph models for social networks. In *Algorithms and models for the web-graph*, pp. 138–149. Springer.

Yu, R., X. He, and Y. Liu (2014). Glad: Group anomaly detection in social media analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 372–381. ACM.

Zaki, M. J. and W. Meira (2013). *Data mining and analysis: fundamental concepts and algorithms*. New York, NY: Cambridge University Press.

Zhao, Y., E. Levina, and J. Zhu (2011). Community extraction for social networks. *Proceedings of the National Academy of Sciences 108*(18), 7321–7326.

Zhao, Y., E. Levina, J. Zhu, et al. (2012). Consistency of community detection in networks under degree-corrected stochastic block models. *The Annals of Statistics 40*(4), 2266–2292.

Zhu, M. and A. Ghodsi (2006). Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis 51*(2), 918–930.

Zoski, K. W. and S. Jurs (1996). An objective counterpart to the visual scree test for factor analysis: The standard error scree. *Educational and Psychological Measurement 56*(3), 443–451.